Advanced Topics in Text Mining

Summer Term 2017: Text Clustering & Topic Modeling

Erich Schubert

Lehrstuhl für Datenbanksysteme,
Institut für Informatik,
Ruprecht-Karls-Universität Heidelberg

Summer Term, 2017, Heidelberg

Preface

This is the print version of the slides of the *first* ATM class in summer 2017.

Some *animations* available in the screen version – in particular in the clustering section – are *condensed into a single slide* to reduce redundancy in the printout and the number of pages.

This class had 11 lecture sessions of 90 minutes each (including time for organizational and QA) plus tutorial sessions with hand-on experience, and gave 4 ECTS points.

Some slides are withheld because of uncertain image copyright.

Organizational slides are not included in this version.
The screen version contained about 172 numbered frames with a total of 361 slides.

Screen version, homework assignments, etc. are currently not available publicly.

This material is made available as-is, with no guarantees on completeness or correctness.

Changes for future versions

The following changes are suggested for a future iteration:

- Reduce: word2vec in Foundations, as there is a separate chapter on word and document embeddings. Initially it was not clear if we will be able to cover this in this class.
- Add: *intrinsic* dimensionality to the curse of dimensionality (which received unexpected interest by the attendees)
- Add: a topic modeling on book title example / homework
- Add: Collection frequency vs. document frequency
- Add: discuss n-gram in the preprocessing.
- Add: Evaluation of IR, e.g., NDCG, Perplexity?
- Add: discussion of PMI, PPMI, in word2vec etc.?
- Add: nonnegative matrix factorization (NMF)
- Add: computation examples with word2vec?

Why is Text Mining Difficult?

Example: Homonyms

Apples have become more expensive.

- Apple computers?
- Apple fruit?

Many homonyms:

- Bayern: the state of Bavaria, or the soccer club FC Bayern?
- Word: the Microsoft product, or the linguistic unit?
- Jam: traffic jam, or jelly?
- A duck, or to duck? A bat, or to bat?
- Light: referring to brightness, or to weight?

Why is Text Mining Difficult?

Example: Negation, sarcasm and irony

This phone may be great, but I fail to see why.

- This actor has never been so entertaining.
- The least offensive way possible
- Colloquial: [. . .] is the shit!
- Sarcasm: Tell me something I don’t know.
- Irony: This cushion is soft like a brick.

Why is Text Mining Difficult?

Example: Errors, mistakes, abbreviations

People are lazy and make mistakes, in particular in social media.

- Let’s eat, grandma. (German: Komm, wir essen, Oma.)
- I like cooking, my family, and my pets.
- *They’re there* with *their* books.
- *You’re* going too fast with *your* car.

- I need food. I am so hungary.
- Let’s grab some bear.

- Next time u r on fb check ur events.
- I’m hangry. (Hungry + angry = hangry)

Recent success is impressive, but also has limits

We cannot "learn" everything

We have seen some major recent successes:

- AI assistants like Google Assistant, Siri, Cortana, Alexa.
- Machine translation like Google, Skype.

But that does not mean this approach works everywhere.

- Require massive training data.
- Require labeled data.
- Most functionality is command based, fallback to web search
  E.g. "take a selfie" is a defined command, and not "understood" by the KI.

For example machine translation: the EU translates millions of pages per year, much of which is publicly available for training translation systems.

Unsupervised text mining—the focus of this class—is much harder!

Why is Text Mining Difficult?

Example: Stanford CoreNLP

Stanford CoreNLP: *The* standard solution for Natural Language Processing (NLP) [Man+14].

NLP is still hard, even just sentence splitting:

Example (from a song list):
```
All About That Bass
by Scott Bradlee's Postmodern Jukebox feat. Kate Davis
```

Sentence 1: `All About That Bass by Scott Bradlee's Postmodern Jukebox feat.`
Sentence 2: `Kate Davis`
Named entity: `Postmodern Jukebox feat`

Best accuracy: 97% on *news* (and $<90\%$ on other text [HEZ15]) $\Rightarrow$ several errors per document!

Many more, and even worse in German (e.g. splits `1. Bundesliga` into two sentences!)

Why is Text Mining Difficult?

Example: Never-Ending Language Learner

Never-Ending Language Learner (NELL) [Mit+15]:

- learning computer system
- reading the web 24 hours/day since January 2010
- knowledge base with over 80 million confidence-weighted beliefs

What NELL believes to know about apple (plant): (with high confidence!)

- steve is a subpart of apple (plant)
- jobs is a subpart of apple (plant)
- steve_jobs is a subpart of apple (plant)
- jobs is a subpart of apple (plant)
- apple has acquired quattro_wireless (company)
- apple is a company also known as apple (bank)
- apple is a company also known as apple (bank)
- apple is a company also known as apple (biotechcompany)

"steve" is believed to be a Canadian journalist.
The first "jobs" is a mixture of Steve Jobs, Steve Wozniak, and Steve Ensminger (LSU football coach) and some other Steve with a wife named Sarah? (The CEO of Apple Bank is Steven Bush)
"steve_jobs" is believed to be a professor and the CEO of "macworld (publication)".
The second "jobs" is a building located in the city vegas.

References I

[HEZ15] T. Horsmann, N. Erbs, and T. Zesch. "Fast or Accurate? - A Comparative Evaluation of PoS Tagging Models". In: *German Society for Computational Linguistics and Language Technology, GSCL*. 2015, pp. 22–30.

[Man+14] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. "The Stanford CoreNLP Natural Language Processing Toolkit". In: *ACL System Demonstrations*. 2014.

[Mit+15] T. M. Mitchell, W. W. Cohen, E. R. H. Jr., P. P. Talukdar, J. Betteridge, A. Carlson, B. D. Mishra, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. A. Platanios, A. Ritter, M. Samadi, B. Settles, R. C. Wang, D. T. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. "Never-Ending Learning". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. 2015, pp. 2302–2310.

Linear Algebra

We will usually be representing documents as vectors!

- Vector space math
  - Vectors
  - Matrices
  - Multiplication
  - Transpose
  - Inverse
- Matrix factorization
  - Principal Component Analysis (PCA, "Hauptachsentransformation"), Eigenvectors and Eigenvalues, …
  - Singular Value Decomposition

$$M = U \Sigma V^T$$

Linear Algebra II

We will omit the $\vec{}$ and ^{T} where they can be easily inferred.

Vector:

$$\vec{v} = (v_1, v_2, \ldots, v_j)^T$$

Matrix:

$$M = \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1j} \\ m_{21} & m_{22} & \cdots & m_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ m_{i1} & m_{i2} & \cdots & m_{ij} \end{pmatrix} = \begin{pmatrix} \vec{m_1}^T \\ \vec{m_2}^T \\ \vdots \\ \vec{m_i}^T \end{pmatrix} = (\vec{m_1}, \vec{m_2}, \ldots \vec{m_i})^T$$

Transpose laws:

$$(M\vec{x})^T = \vec{x}^T M^T$$

Orthogonality:

$$O^T = O^{-1}$$

Computer scientists usually store vectors in rows, not columns. So notation *will* vary across sources. Double-check dimensions every time.

Linear Algebra III – PCA

On *centered* (or standardized) data, with weights $\sum_i \omega_i = 1$, we have

$$X^T X = \sum_i \omega_i v_i v_i^T = \mathrm{Cov}(X)$$

Decompose this Covariance matrix into:

$$X^T X = W^T \Sigma^2 W = (\Sigma W)^T \ \Sigma W$$

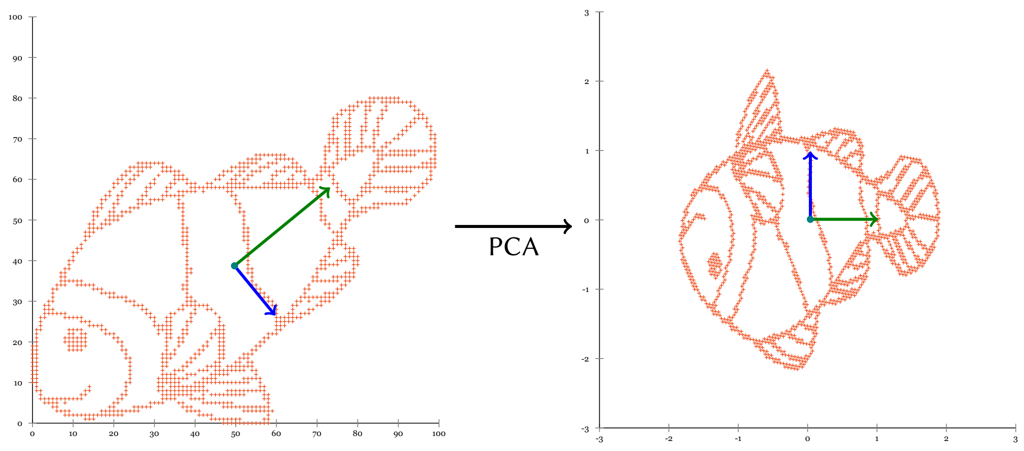
where W is an orthonormal (rotation) matrix, and Σ is a diagonal (scaling) matrix.

The vectors of W are called eigenvectors, the values of Σ are called eigenvalues.

Project using:

$$x' := \underbrace{\Sigma}_{\text{Scale}} \underbrace{W}_{\text{Rotate}} x$$

Linear Algebra IV – PCA II



Linear Algebra V – SVD

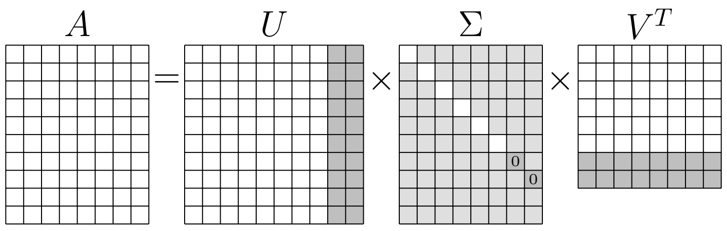
The decomposition in PCA is usually implemented using the SVD routine.

Singular Value Decomposition is a more general decomposition procedure.
It allows us to decompose any $m \times n$ matrix into

$$A = U \Sigma V^T$$

where U is an orthogonal $m \times m$ matrix,
Σ is a diagonal $m \times n$ matrix,
V is an orthogonal $n \times n$ matrix.
By convention, Σ is arranged by descending values.

Linear Algebra VI – SVD II



At first sight, this makes the data even larger. So why do we want to do this?

- Matrix properties are beneficial: orthogonal (U, V) respectively diagonal (Σ).
- If Σ has zeros on the diagonal, we can reduce the matrix sizes *without loss*.
- We can approximate (with least-squared error) the data by further shrinking the matrix.

Intuition: U maps rows to factors, Σ is the factor weight, and V maps factors to columns.
PCA is often used the same way – by keeping only the most important components!

Statistics

We are often using statistical language models!

- Elementary probability theory
- Conditional probabilities

$$P(A|B)$$

- Bayes' rule

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

- Random variables, probability density, cumulative density
- Expectation and variance.

Statistics I – Probabilities

Some basic terms and properties:

- $P(A)$: Probabilitiy that A occurs
- $P(AB) = P(A \wedge B) = P(A \cap B) = P(A, B)$: Probability that A and B occur
- $P(A|B) = \frac{P(A \wedge B)}{P(B)}$: Conditional probability that A occurs *if* B occurs
- $P(A \vee B) = P(A \cup B) = P(A) + P(B) - P(AB)$: A or B (or both) occur
- $P(A \cap B) = P(A) \cdot P(B) \Leftrightarrow$: A and B are independent
- $0 \leq P(A) \leq 1$; $P(\emptyset) = 0$; $P(\neg A) = 1 - P(A)$

Statistics II – Bayes' rule

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$
$$\Rightarrow P(A \wedge B) = P(A|B) \cdot P(B)$$
$$\Rightarrow P(B \wedge A) = P(B|A) \cdot P(A)$$

Because $A \wedge B = B \wedge A$, these are equal:

$$\Rightarrow P(A|B)P(B) = P(B|A)P(A)$$

And we can derive Bayes' rule:

$$\Rightarrow P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

Statistics III — Bayes' rule II

$$P(B|A) = P(A|B)\frac{P(B)}{P(A)}$$

Why is Bayes' rule important?Bayes' rule is important, because it allows us to reason "backwards".

If we know the probability of $B \to A$, we can compute the probability of $A \to B$.

If one of these values cannot be observed, we can estimate it using Bayes' rule.

If we do not know either $P(A)$ or $P(B)$, we may still be able to cancel it out it some equations (if we can look at the relative likelihood of two complementary options, e.g., spam vs. not spam).

Statistics IV — Bayes' rule III

Example – Breast Cancer detection:

	Probability	Test positive	Test negative
Breast cancer	1%	80%	20%
No breast cancer	99%	9.6%	90.4%

Question: is this a good test for breast cancer?Naive answer: Test results are 80–90% correct.

If the test result is positive, what is the probability of having cancer (= test is correct)?

$$\begin{aligned} P(\text{cancer}|\text{positive}) &= P(\text{positive}|\text{cancer}) \cdot \frac{P(\text{cancer})}{P(\text{positive})} \\ &= 80\% \cdot \frac{1\%}{80\% \cdot 1\% + 9.6\% \cdot 99\%} \\ &\approx 7.8\% \end{aligned}$$

In $> 90\%$ of "cancer detected" cases, the patient is fine! (Because 10% of 99% $\gg$ 80% of 1%.)

Statistics V – Random Variables and Probability Density

Random variable X: maps outcomes to some measureable quantity (typically: real value).

Example: $\underbrace{\text{throw of acutal dices on the table}}_{\text{Unique event}} \mapsto \underbrace{\text{sum of eyes on dices}}_{\text{Variable that we model}}$

Variables are often binary (head=1, tail=0), discrete (2…12 eyes), or real valued.

Discrete:

Probability mass function: $\operatorname{pmf}_X(x_i) = P(X = x_i)$

Continuous (real valued):

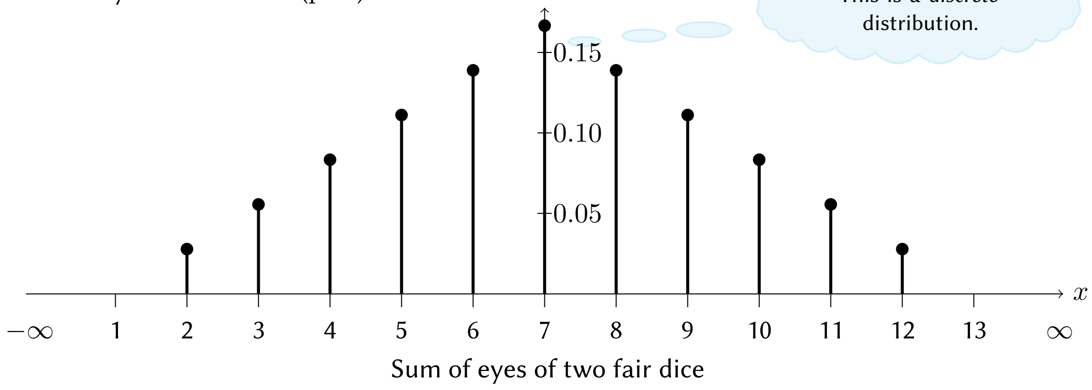
Probability density function: $\operatorname{pdf}_X(x) = \frac{\mathrm{d}}{\mathrm{d}x} \operatorname{cdf}_X(x)$

Cumulative density function: $\operatorname{cdf}_X(x) = P(X \leq x) \quad (\operatorname{cdf}_X(x) = \int_{-\infty}^{x} \operatorname{pdf}_X(x)\,\mathrm{d}x)$

Notes: The point probability $P(X = x)$ of a continuous variable is usually 0, because there is an infinite number of real numbers – consider the probability of measuring a temperature of exactly π: $P(\text{Temperature} = \pi)$.
The $\operatorname{cdf}(x)$ exists for discrete and continuous, but is less commonly used with discrete variables.
The $\operatorname{pdf}(x)$ can be larger than 1, and is *not* a probability (the cdf and pmf are)!

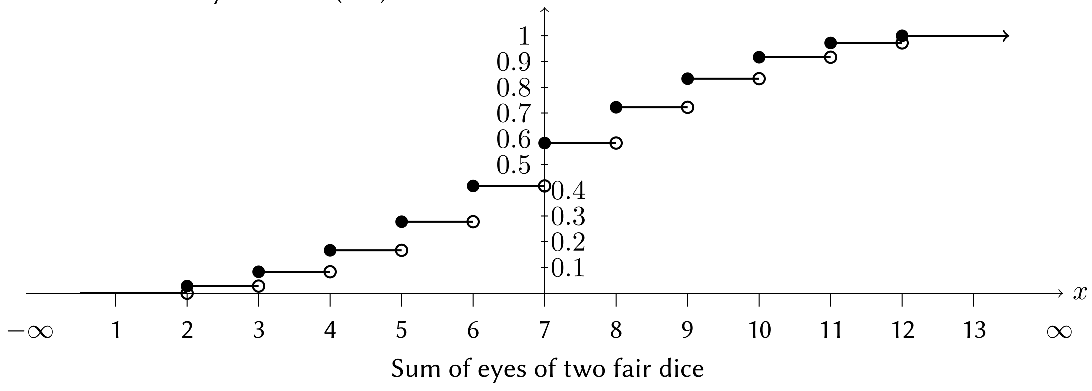
Statistics V – Random Variables and Probability Density

Probability mass function (pmf):



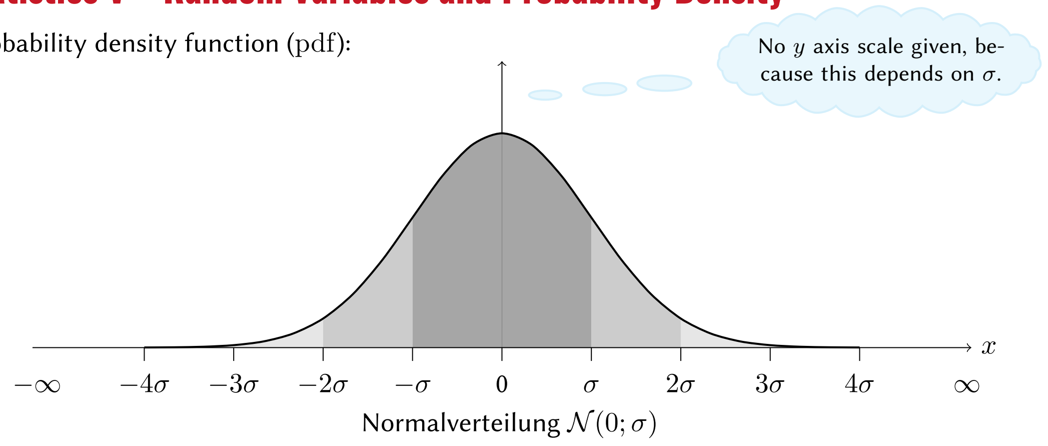
Statistics V – Random Variables and Probability Density

Cumulative density function (cdf):



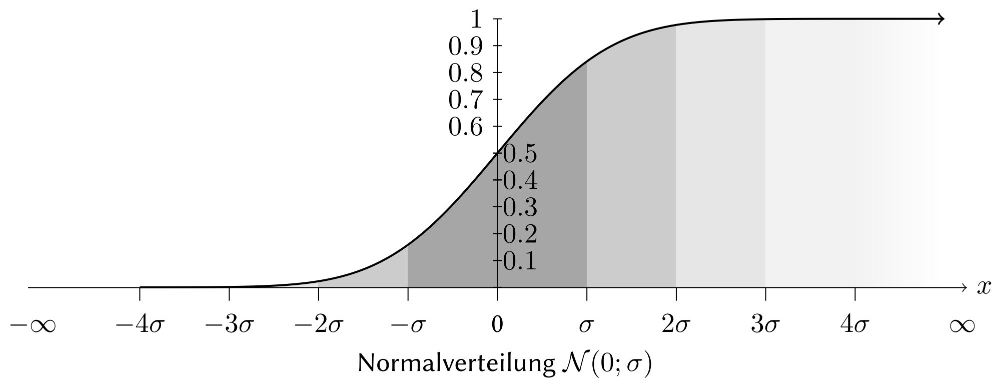
Statistics V – Random Variables and Probability Density

Probability density function (pdf):



Statistics V – Random Variables and Probability Density

Cumulative density function (cdf):



Statistics VI – Expectation and Variance

Expected value (discrete):

$$\mathrm{E}[X] = \mu_X = \sum_i p_i x_i = \sum_i \mathrm{pmf}_X(x_i) x_i$$

Expected value (continuous):

$$\mathrm{E}[X] = \mu_X = \int_{-\infty}^{\infty} \mathrm{pdf}_X(x) x \, \mathrm{d}x$$

Variance:

$$\mathrm{Var}[X] = \sigma_X^2 = \mathrm{E}\left[(X - \mathrm{E}[X])^2 \right]$$

Useful for proofs, but problematic with floating point numerics:

$$\mathrm{Var}[X] = E[X^2] - E[X]^2$$

σ is called the standard deviation.

Information Theory

In information theory, logarithms are usually base 2.

Data mining is all about *information*!

- Shannon Entropy

$$H = -\sum_i p_i \log p_i$$

- Mutual Information

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

- Kullback-Leibler Divergence

$$KL(P|Q) = \sum_i p_i \log \frac{p_i}{q_i}$$

Data Mining

Text mining is data mining applied to text!

- Classification
  - Support Vector Machines & Kernel Trick
  - Nearest-Neighbor classification
  - Naive Bayes
- Cluster analysis
  - Hierarchical clustering
  - k-means clustering
- Frequent Itemset Mining
  - APRIORI, Eclat, FPgrowth

We will summarize this as necessary in the lecture, but it is best if you already know the basics.

Lexical Units

A tiny bit of linguistics

Terminology we may use at some point in the lecture:

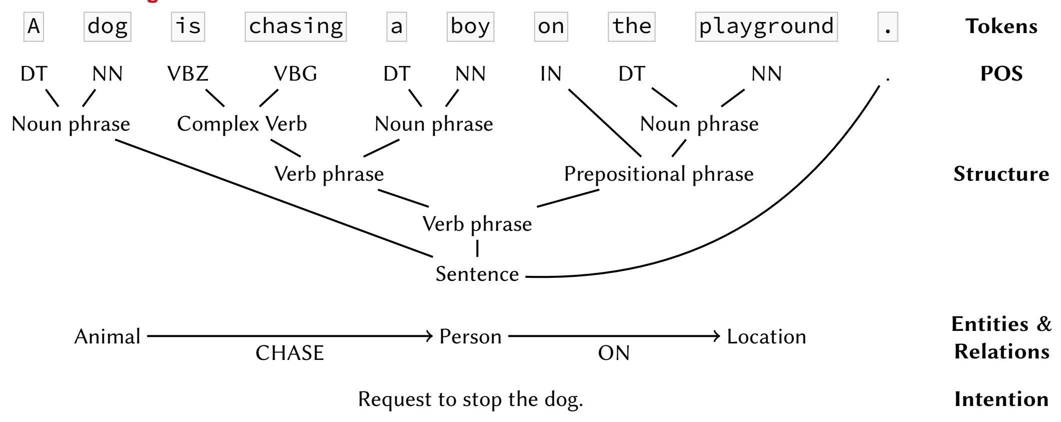
- A *document* is a longer piece of text.
- A *section* is a logical section within a document.
- A *sentence* is a sequence of tokens in a section, usually ending with a dot.[1]
- A *token* usually is a word, but can also be, e.g., interpunction.
- A *phrase* is a short sequence of tokens (part of a sentence).
- A *stem* is the prefix of a word with inflection endings removed (e.g. fishing $\rightarrow$ fish).
- A *lemma* is the logical base form (e.g., better $\rightarrow$ good).[2]
- A *POS-tag* is the part-of-speech interpretation of a token, e.g., verb, or noun.

[1]At the end of a logical section—e.g., a headline—the dot may be missing

[2]The noun “a meeting” and the verb “to meet” (e.g. in “we are meeting”) are different lemmata.

Part of speech and syntactic structure

A little bit of linguistics



Example taken from ChengXiang Zhai [ZM16]

Motivation

Why we want to represent data as vectors.

We have text, i.e., a sequence of characters.
Many algorithms expect a vector from $\mathbb{R}^d$.

➡ We need to convert text to vectors.

Text = bytes = vector? ASCII: A=65, B=66, C=67, D=68, E=69, …
`Example = [69, 120, 97, 109, 112, 108, 101]?`

This does not work well at all!

Algorithms assume that for every dimension i: $x_i \sim y_i$ if x and y are similar.
But documents can be similar, yet different in almost every byte position:

```
He said: This is an example. 
She said: This is an example.
```

➡ We need a vector space with meaningful *positions*.

Bag of Words

Vectorizing Text

A “bag” or “multiset” is a set that can contain multiple instances of the same element

Separate the documents into words, discard word order:

Birds of a feather flock together .
$\rightarrow$ a bird feather flock of together

It is the early bird that gets the worm .
$\rightarrow$ bird early get is it that the$\times 2$ worm

But the second mouse gets the cheese .
$\rightarrow$ but cheese get mouse second the$\times 2$

Early to bed and early to rise , makes a man healthy , wealthy and wise .
$\rightarrow$ a and$\times 2$ bed early$\times 2$ healthy make man rise to$\times 2$ wealthy wise

For better results, normalize words: birds $\rightarrow$ bird, gets $\rightarrow$ get. Discard . and ,.

Bag of Words II

Term-Document Matrix

Dim	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
	`a`	`and`	`bed`	`bird`	`but`	`cheese`	`early`	`feather`	`flock`	`get`	`healthy`	`is`	`it`	`make`	`man`	`mouse`	`rise`	`of`	`second`	`that`	`the`	`to`	`together`	`wealthy`	`wise`	`worm`
Doc 1	1			1				1	1									1					1			
Doc 2				1			1			1		1	1							1	2					1
Doc 3					1	1				1						1			1		2					
Doc 4	1	2	1				2				1			1	1		1					2		1	1	

Denote as:
$\mathrm{tf}_{\mathrm{to,Doc\ 4}} = 2$

Note: for a large document collection, we will have thousands of dimensions!

Do not *store* 0s
Use sparse data!

Similar documents should now have similar vectors.

➡ How can we measure similarity?

Bag of Words III

TF-IDF

Words such as `a`, `and`, `but`, `is`, `it`, `of`, `that`, `the` `to` are not helpful for differentiating documents. We could remove them ($\rightarrow$ stopword removal), or we assign them a low weight.

TF-IDF (Term frequency $\times$ inverse document frequency) is a popular solution to this.[3]

TF: Term Frequency
IDF: Inverse Document Frequency

$$\mathrm{idf}_t := \log \frac{N}{\mathrm{df}_t} = -\log \frac{\mathrm{df}_t}{N} = -\log P(t \in d)$$

(where $N = |D|$ is the number of documents,
and $\mathrm{df}_t = |\{d \in D \wedge t \in d\}|$ is the document frequency – number of documents with term t)

This weight is similar to Shannon information. More informative terms have more weight.
But the theoretical justification is difficult [Spä72; Spä73; RS76; SWR00a; SWR00b; Rob04].

[3]TF-IDF is also written as "tf.idf" and "tf$\times$idf". This is not a minus, but a hyphen; we always use multiplication.

Bag of Words IV

TF-IDF variations

Increased importance of repeated words

Common terms get less weight

Normalize for document length

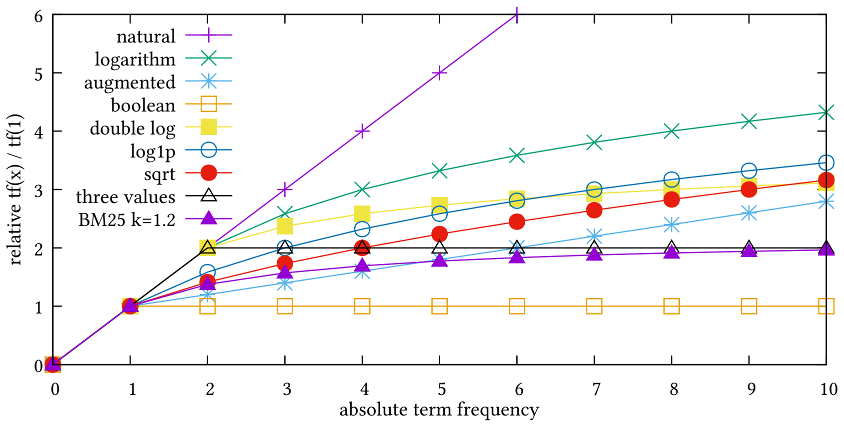
There exist many variations of TF-IDF (in SMART notation [Sal71; SB88]): [MRS08]

Term frequency (if $\mathrm{tf}_{t,d} > 0$)		Document frequency		Document Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log \mathrm{tf}_{t,d}$	t (idf)	$\log N / \mathrm{df}_t$	c (cosine)	$1 \Big/ \sqrt{\sum_i w_i^2}$
a (augmented)	$0.5 + \frac{0.5 \cdot \mathrm{tf}_{t,d}}{\max_t \mathrm{tf}_{t,d}}$	p (prob idf)	$\max\{0, \log \frac{N - \mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$ (see [MRS08])
b (boolean)	1	(idf smooth)	$\log N / (\mathrm{df}_t + 1)$	b (byte size)	$1/\mathrm{CharLength}^\alpha, \alpha < 1$
L (log mean)	$\frac{1 + \log \mathrm{tf}_{t,d}}{1 + \mathrm{mean}_{t \in d} \log \mathrm{tf}_{t,d}}$				
d (double log)	$1 + \log(1 + \log \mathrm{tf}_{t,d})$				
(BM25)	$\frac{k \cdot \mathrm{tf}_{t,d}}{k + b \cdot (L-1) + \mathrm{tf}_{t,d}}$	(BM25)	$\log \frac{N - \mathrm{df}_t + .5}{\mathrm{df}_t + .5}$		
(three val.)	$\min\{\mathrm{tf}_{t,d}, 2\}$				
(log1p)	$\log(1 + \mathrm{tf}_{t,d})$				
(sqrt)	$\sqrt{\mathrm{tf}_{t,d}}$			(manhattan)	$1/\sum_i w_i$

Xapian and Lucene search now default to the BM25 variant [RZ09].

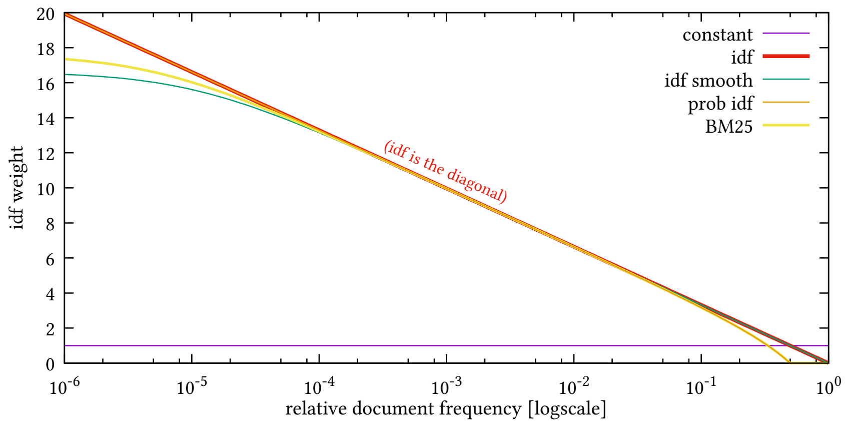
Bag of Words V

TF and IDF variations visualized



Bag of Words V

TF and IDF variations visualized



Exact Text Search

Binary Retrieval Model

If we want to search for `bird` + `early`, this corresponds to using *AND* on these columns:

Dim	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
	`a`	`and`	`bed`	`bird`	`but`	`cheese`	`early`	`feather`	`flock`	`get`	`healthy`	`is`	`it`	`make`	`man`	`mouse`	`rise`	`of`	`second`	`that`	`the`	`to`	`together`	`wealthy`	`wise`	`worm`
Doc 1	1			1				1	1									1					1			
Doc 2				1			1			1		1	1							1	1					1
Doc 3					1	1				1						1			1		1					
Doc 4	1	1	1				1				1			1	1		1					1		1	1	

`bird` AND `early` = 1100 AND 0101 = 0100
Matching document: Bit 2 = Document 2.

Exact Text Search II

Inverted Index

We cannot store the binary matrix for large document collections.

Solution: We store only the 1's, *for each term.*

`bird` $\rightarrow$ `Doc 1`, `Doc 2`
`early` $\rightarrow$ `Doc 2`, `Doc 4`

Optimizations:

- Sort lists, and use a *merge* operation to intersect lists in $O(n+m)$
- Skip pointers to skip multiple entries at once.
- B-trees with prefix compression to organize lists.
- Compress lists by storing deltas, variable-length integer encoding etc. [LB15; LKK16] (Reduce IO cost, use SIMD instructions for fast decoding and intersection.)

Exact Text Search III

Inverted Index

We can store auxiliary information in inverted indexes:

Auxiliary information	Symbol	Cost
Total number of documents	df_t	$O(\lvert \mathrm{Terms} \rvert)$
Number of occurrences	$\mathrm{tf}_{t,d}$	$O(N \cdot \text{avg-length})$
Total number of occurrences	$\sum_d \mathrm{tf}_{t,d}$	$O(\lvert \mathrm{Terms} \rvert)$
Maximum number of occurrences	$\max_d \mathrm{tf}_{t,d}$	$O(\lvert \mathrm{Terms} \rvert)$
Term positions within document		$O(N \cdot \text{avg-length})$

(for phrase matches and the "NEAR" operator – usually 2–4$\times$ larger index)

For TF-IDF, we can store the quantity/position along with the document:

`bird` $\rightarrow$ `Doc 1`$\times 1$, `Doc 2`$\times 1$
`early` $\rightarrow$ `Doc 2`$\times 1$, `Doc 4`$\times 2$

`early` $\rightarrow$ `Doc 2` @ 4, `Doc 4` @ 1,5

Ranked Retrieval

Approximative Matching

Exact matches (boolean model) tend to return no results, or too many results.

➨ We often want to see the "best" matches only.

We need a *scoring function* to sort results.

Intuition: TF: the more words match in the document, the better.
Intuition: IDF: common words should be given less weight than rare words.

Cosine similarity:

$$\cos(A,B) := \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2} \cdot \sqrt{\sum_i b_i^2}}$$

where A, B are the TF-IDF vectors.

Note: with `c` normalization, $\cos(A,B) \underset{\substack{\|A\|=1 \\ \|B\|=1}}{=} A \cdot B = \sum_i a_i b_i$.

Ranked Retrieval II

Approximative Matching with TF-IDF

For text search (unweighted query Q), we can simplify this to:

$$score(Q, D) := \sum_{t \in Q} \text{tf-idf}_{t,d}$$

This has the benefits:

- Efficient computation, one term at a time, and Q is usually small.
- Documents *not* in the inverted list get $+0$ (i.e. no change).

Improvements:

- Order terms $t \in Q$ by descending idf_t, and try to stop early if the remaining documents cannot become a top-k result anymore (if we know $\max_d \text{tf-idf}_{t,d}$).
- Stop early and skip low-idf terms, *even* if we cannot guarantee the result to be correct (the similarity it is only an approximation of real relevance anyway – it is never "correct", and frequent low-idf terms such as "in" and "the" will not change much anyway.)

Ranked Retrieval III

TF-IDF Similarity for Clustering

For clustering etc. we do not want such a query-document asymmetry.
So we will usually use cosine similarity. If we normalize documents, this can be computed as the *dot product* of the TF-IDF vectors ($\text{tf-idf}_A \cdot \text{tf-idf}_B$):

$$\cos(A, B) \underset{\substack{\|A\|=1 \\ \|B\|=1}}{=} \sum_t \text{tf-idf}_{t,A} \cdot \text{tf-idf}_{t,B}$$

Since $t \notin A \wedge t \notin B \Rightarrow \text{tf}_{t,A} = 0 \wedge \text{tf}_{t,B} = 0 \Rightarrow \text{tf-idf}_{t,A} \cdot \text{tf-idf}_{t,B} = 0$, we only need:

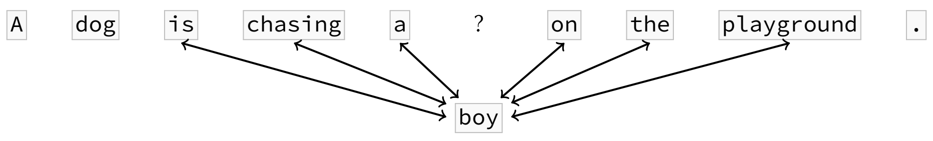
$$\cos(A, B) \underset{\substack{\|A\|=1 \\ \|B\|=1}}{=} \sum_{t \in A \cap B} \text{tf-idf}_{t,A} \cdot \text{tf-idf}_{t,B}$$

Note: $t \in A \cap B$ is usually a small set if the documents are dissimilar.

Do *not* materialize $A \cap B$ – compute the dot product by *merging* the (sorted) sparse vectors, skipping all t where $t \notin A \wedge t \notin B$ (optimized sparse vector product).

Contextual Word Representations

Word Context



- The context of a word is representative of the word.
- Similar words often have a similar context (e.g., `girl`).
- Statistics can often predict the word, based on the context.
- Context of a word ≈ a document: `a`, `chasing`, `is`, `on`, `playground`, `the`

➡ Try to model *words* based on their context

But: many documents per word, same problems as with real documents, ...

Contextual Word Representations II

Alternatives to Bag-of-Words

Can we *learn* a $\mathbb{R}^d$ representation of words/text? [RHW86; Ben+03; RGP06]
(Recently: word2vec [Mik+13], vLBL [MK13], HPCA [LC14], [LG14a], GloVe [PSM14])

Basic idea:

1. Train a neural network (a map function, factorize a matrix) to either:
   - predict a word, given the preceding and following words (Continuous Bag of Words, CBOW)
   - predict the preceding and following words, given a word (Skip-Gram)
2. Configure one layer of the network to have d dimensions (for small d)
   Usually: one layer network (not *deep*), 100 to 1000 dimensions.
3. Map every word to this layer, and use this as feature.

Note: this maps words, not documents!

We can treat the document ID like a word, and map it the same way. [LM14]

Word2Vec

Beware of cherry picking

Famous example (with the famous “Google News” model):

Berlin	is to	Germany	as	Paris	is to	➡	France
Berlin	−	Germany	=	Paris	−		France

Beware of cherry picking!

Berlin	is to	Germany	as	Washington_D.C.	is to	➡	Spending_Surges
Ottawa	is to	Canada	as	Washington_D.C.	is to	➡	Quake_Damage
Germany	is to	Berlin	as	United_States	is to	➡	U.S.
Apple	is to	Microsoft	as	Volkswagen	is to	➡	VW
man	is to	king	as	boy	is to	➡	kings
king	is to	man	as	prince	is to	➡	woman

Computed using `https://rare-technologies.com/word2vec-tutorial/`

Word2Vec II

Beware of data bias

Context of Munich in Reuters News!

Most similar words to `Munich`:
`Munich_Germany`, `Dusseldorf`, `Berlin`, `Cologne`, `Puchheim_westward`
➡ Many stock photos with “Puchheim westward of Munich”, used in gas price articles.

Most similar words to `Berlin`:
`Munich`, `BBC_Tristana_Moore`, `Hamburg`, `Frankfurt`, `Germany`
➡ Tristana Moore is a key BBC correspondent in Berlin.

Most similar words to `Heidelberg`:
`CEO_Bernhard_Schreier`, `CFO_Dirk_Kaliebe`, `Würzburg`, `Heidleberg`, `Heidelberger_Druckmaschinen_AG`
➡ CEO, CFO of Heidelberger Druckmaschinen. Würzburg – because of Koenig & Bauer?

Computed using `https://rare-technologies.com/word2vec-tutorial/`

Word2Vec and Word Embeddings

Strengths & Limitations

- Focused primarily on words, not on documents
- Captures certain word semantics surprisingly well
- Mostly preseves linguistic relations: plural, gender, language, …
  (And thus very useful for machine *translation*)
- Requires massive training data
  (Needs to learn projection matrixes of size $N \times d$)
- Only works for frequent-enough words, unreliable on low-frequency words
- Does not distinguish homonyms, and is affected by training data bias
- How to fix, if it does not work as desired?
- Not very well understood yet [GL14; LG14b; LGD15], but related to matrix factorization [PSM14]

Summary

- We often need a $\mathbb{R}^d$ representation of documents.
- Text needs to be tokenized, maybe NLP analysis.
- Sparse representation allows using text search techniques.
- Similarity is often measured by Cosine (on sparse representations).
- TF-IDF normalization improves search and similarity results.
- Many heuristic choices (e.g., TF-IDF variant)
- Dense models (e.g., word2vec) are a recent hype
  But: need huge training data, no guarantees, hard to fix if they do not work right, mostly used for single word similarity and machine translation.

Literature

Recommended literature:

- Vector space model and information retrieval:
  Chapter 6 “Scoring, term weighting and the vector space model”
  Chapter 7 “Computing scores in a complete search system”
  Chapter 11 “Probabilistic information retrieval“

  C. D. Manning, P. Raghavan, and H. Schütze
  Introduction to information retrieval
  Cambridge University Press, 2008
  ISBN: 978-0-521-86571-5
  URL: `http://nlp.stanford.edu/IR-book/`
- Word embeddings:
  [Ben+03; RGP06; Mik+13; MK13; LM14; GL14; LC14; LG14a; PSM14; LGD15]

References I

[Ben+03] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. "A Neural Probabilistic Language Model". In: *J. Machine Learning Research* 3 (2003), pp. 1137–1155.

[GL14] Y. Goldberg and O. Levy. "word2vec explained: Deriving Mikolov et al.'s negative-sampling word-embedding method". In: *CoRR* abs/1402.3722 (2014).

[LB15] D. Lemire and L. Boytsov. "Decoding billions of integers per second through vectorization". In: *Softw., Pract. Exper.* 45.1 (2015), pp. 1–29.

[LC14] R. Lebret and R. Collobert. "Word Embeddings through Hellinger PCA". In: *European Chapter of the Association for Computational Linguistics, EACL*. 2014, pp. 482–490.

[LG14a] O. Levy and Y. Goldberg. "Linguistic Regularities in Sparse and Explicit Word Representations". In: *Computational Natural Language Learning, CoNLL*. 2014, pp. 171–180.

[LG14b] O. Levy and Y. Goldberg. "Neural Word Embedding as Implicit Matrix Factorization". In: *Neural Information Processing Systems, NIPS*. 2014, pp. 2177–2185.

[LGD15] O. Levy, Y. Goldberg, and I. Dagan. "Improving Distributional Similarity with Lessons Learned from Word Embeddings". In: *TACL* 3 (2015), pp. 211–225.

[LKK16] D. Lemire, G. S. Y. Kai, and O. Kaser. "Consistently faster and smaller compressed bitmaps with Roaring". In: *Softw., Pract. Exper.* 46.11 (2016), pp. 1547–1569.

References II

[LM14] Q. V. Le and T. Mikolov. "Distributed Representations of Sentences and Documents". In: *International Conference on Machine Learning, ICML*. 2014, pp. 1188–1196.

[Mik+13] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* abs/1301.3781 (2013).

[MK13] A. Mnih and K. Kavukcuoglu. "Learning word embeddings efficiently with noise-contrastive estimation". In: *Neural Information Processing Systems, NIPS*. 2013, pp. 2265–2273.

[MRS08] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval.* Cambridge University Press, 2008. ISBN: 978-0-521-86571-5. URL: `http://nlp.stanford.edu/IR-book/`.

[PSM14] J. Pennington, R. Socher, and C. D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing, EMNLP*. 2014, pp. 1532–1543.

[RGP06] D. L. Rohde, L. M. Gonnerman, and D. C. Plaut. "An improved model of semantic similarity based on lexical co-occurrence". self-published. 2006.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (Oct. 1986), pp. 533–536.

[Rob04] S. Robertson. "Understanding inverse document frequency: on theoretical arguments for IDF". In: *Journal of Documentation* 60.5 (2004), pp. 503–520.

[RS76] S. E. Robertson and K. Spärck Jones. "Relevance weighting of search terms". In: *JASIS* 27.3 (1976), pp. 129–146.

References III

[RZ09] S. E. Robertson and H. Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Foundations and Trends in Information Retrieval* 3.4 (2009), pp. 333–389.

[Sal71] G. Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1971.

[SB88] G. Salton and C. Buckley. "Term-Weighting Approaches in Automatic Text Retrieval". In: *Inf. Process. Manage.* 24.5 (1988), pp. 513–523.

[Spä72] K. Spärck Jones. "A statistical interpretation of term specificity and its application in retrieval". In: *Journal of Documentation* 28.1 (1972), pp. 11–21.

[Spä73] K. Spärck Jones. "Index term weighting". In: *Information Storage and Retrieval* 9.11 (1973), pp. 619–633.

[SWR00a] K. Spärck Jones, S. Walker, and S. E. Robertson. "A probabilistic model of information retrieval: development and comparative experiments - Part 1". In: *Inf. Process. Manage.* 36.6 (2000), pp. 779–808.

[SWR00b] K. Spärck Jones, S. Walker, and S. E. Robertson. "A probabilistic model of information retrieval: development and comparative experiments - Part 2". In: *Inf. Process. Manage.* 36.6 (2000), pp. 809–840.

[ZM16] C. Zhai and S. Massung. *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining.* New York, NY, USA: Association for Computing Machinery and Morgan & Claypool, 2016. ISBN: 978-1-97000-117-4.

What is clustering?

Core concepts

Divide data into clusters:

- Clusters *not* defined beforehand (otherwise: use classification)
- Similar objects should be in the same cluster
- Dissimilar objects in different clusters
  - Different notions of (dis-) similarity
- Based on statistical properties such as:
  - Connectivity
  - Separation
  - Least squared deviation
  - Density

What is clustering?

Clustering use examples

Usage examples:

- Customer segmentation:
  Optimize ad targeting or product design for different “focus groups”.
- Web visitor segmentation:
  Optimize web page navigation for different user segments.
- Data aggregation:
  Represent many data points with a single (representative) example.
  E.g., reduce color palette of an image
- Text collection organization:
  Group text documents into (previously unknown) topics.

Clustering algorithms

Different categories of algorithms

Paradigm:

- Distance
- Variance
- Density
- Connectivity
- Probability
- Subgraph

Properties:

- Partitions: strict, hierarchical, overlapping
- Outliers, or total clustering
- Hard (binary) assignment or soft (fuzzy) assignment
- Full dimensional or subspace
- Rectangular, spherical, or correlated

Hierarchical Agglomerative Clustering I

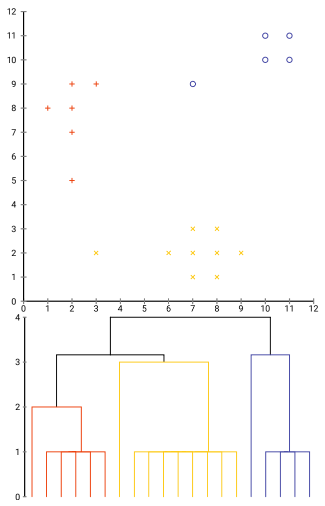
Repeated merging of clusters

One of the earliest clustering methods [Sne57; Sib73; Har75; KR90]:

1. Initially, every object is a cluster
2. Find two most similar clusters, and merge them
3. Repeat (2) until only one cluster remains
4. Plot tree ("dendrogram"), and choose interesting subtrees

Many variations that differ by:

- Distance / similarity measure of *objects*
- Distance measure of clusters ("linkage")
- Optimizations



Hierarchical Agglomerative Clustering II

Distance of objects

$\|x\|$ usually refers to the Euclidean norm.

We first need distances of *single* objects.
Both Euclidean distance (the most common distance we use):

$$d_{\text{Euclidean}}(x,y) = \sqrt{\sum\nolimits_d (x_d - y_d)^2} \qquad = \|x-y\|_2$$

and Manhattan distance (city block metric):

$$d_{\text{Manhattan}}(x,y) = \sum\nolimits_d |x_d - y_d| \qquad = \|x-y\|_1$$

are special cases of Minkowski norms (L_p distances):

$$d_{L_p}(x,y) = \left(\sum\nolimits_d |x_d - y_d|^p\right)^{1/p} = \|x-y\|_p$$

➡ Many more distance functions [DD09]!

Hierarchical Agglomerative Clustering III

Distance of objects II

Instead of distances, we can also use similarities:

Cosine similarity:

$$\cos(X,Y) := \frac{X \cdot Y}{\|X\| \cdot \|Y\|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \cdot \sqrt{\sum_i y_i^2}}$$

on L_2 normalized data, this simplifies to:

$$\cos(X,Y) \underset{\substack{\|X\|=1\\ \|Y\|=1}}{:=} X \cdot Y = \sum_i x_i y_i$$

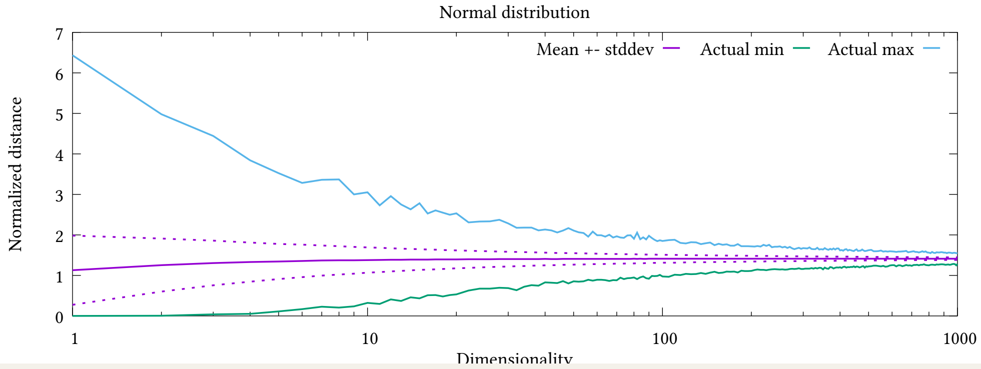
➡ Careful: if we use similarities, large values are better
– with distances, small values are better.

Curse of Dimensionality

Concentration of Distances

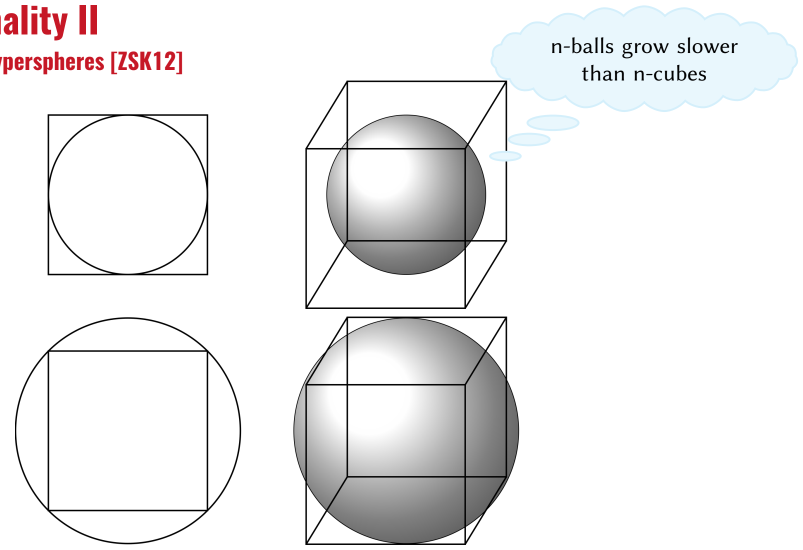
Curse of Dimensionality of Beyer et al. [Bey+99]

$$\text{If } \lim_{d\to\infty} \operatorname{Var}\left(\frac{\|X_d\|}{E[\|X_d\|]}\right) = 0, \text{ then } \frac{D_{\max} - D_{\min}}{D_{\min}} \to 0.$$



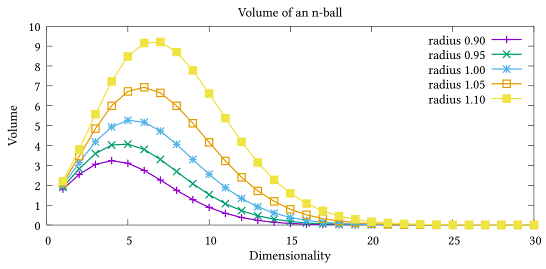
Curse of Dimensionality II

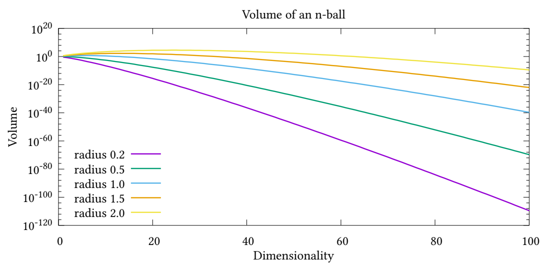
Illustration: “shrinking” (?) hyperspheres [ZSK12]



Curse of Dimensionality III

Illustration: “shrinking” (?) hyperspheres II [ZSK12]





$$V_n(R) = \frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}+1\right)} R^n$$

Curse of Dimensionality IV

Summary

Due to the Curse of Dimensionality, *distances become very similar.*

- Usually at around 10–50 dimensions, this becomes a problem.
- Text usually has 10000+ dimensions (but mostly 0s – sparse)
  Intrinsic dimensionality of text is still high!
- While some claim "cosine is better in high dimensionality" this is false
  (because cosine ≅ Euclidean on the unit sphere)
- The signal-to-noise-ratio is essential [ZSK12]
- In text, we (naturally) have a lot of noise, because of typos and discrete input!
- Rankings can be more meaningful than scores [Hou+10]
- The Curse has many different forms [ZSK12]

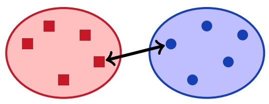
$$\cos(X,Y) \underset{\substack{\|X\|=1\\\|Y\|=1}}{=} X\cdot Y = \sum\nolimits_i \underbrace{x_i y_i}_{\pm \text{ many tiny errors = a lot of noise}}$$

Hierarchical Agglomerative Clustering IV

Distance of clusters I

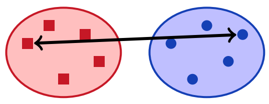
Single-linkage: minimum distance ≅ maximum similarity

$$d_{\text{single}}(A,B) := \min_{a\in A, b\in B} d(a,b) \cong \max_{a\in A, b\in B} s(a,b)$$



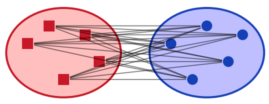
Complete-linkage: maximum distance ≅ minimum similarity

$$d_{\text{complete}}(A,B) := \max_{a\in A, b\in B} d(a,b) \cong \min_{a\in A, b\in B} s(a,b)$$



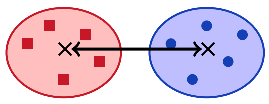
Average-linkage (UPGMA): average distance ≅ average similarity

$$d_{\text{average}}(A,B) := \frac{1}{|A|\cdot|B|} \sum\nolimits_{a\in A} \sum\nolimits_{b\in B} d(a,b)$$



Centroid-linkage: distance of cluster centers (Euclidean only)

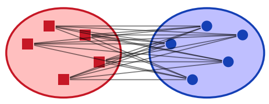
$$d_{\text{centroid}}(A,B) := \|\mu_A - \mu_B\|^2$$



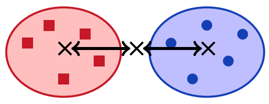
Hierarchical Agglomerative Clustering V

Distance of clusters II

McQuitty (WPGMA): average of previous sub-clusters
Defined recursively, e.g., via Lance-Williams equation.
Average distance to the previous two clusters.

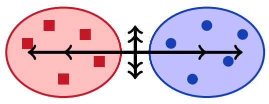


Median-linkage (Euclidean only): distance from midpoint
Defined recursively, e.g., via Lance-Williams equation.
Median is the halfway point of the previous merge.



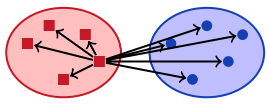
Ward-linkage (Euclidean only): Minimum increase of squared error

$$d_{\text{Ward}}(A,B) := = \frac{|A|\cdot|B|}{|A\cup B|} \|\mu_A - \mu_B\|^2$$



Mini-Max-linkage: Best maximum distance, best minimum similarity

$$d_{\text{minimax}}(A,B) := \min_{c\in A\cup B} \max_{p\in A\cup B} d(c,p) \cong \max_{c\in A\cup B} \min_{p\in A\cup B} s(c,p)$$



Hierarchical Agglomerative Clustering VI

AGNES – Agglomerative Nesting [KR90]

AGNES, using the Lance-Williams equations [LW67]:

1. Compute the pairwise distance matrix of objects
2. Find position of the minimum distance $d(i, j)$ (similarity: maximum similarity $s(i, j)$)
3. Combine rows and columns of i and j into one using Lance-Williams update equations
$$d(A \cup B, C) = \text{LanceWilliams}\,(d(A, C), d(B, C), d(A, B))$$
using only the *stored, known* distances $d(A, C)$, $d(B, C)$, $d(A, B)$.
4. Repeat from (2.) until only one entry remains
5. Return dendrogram tree

Hierarchical Agglomerative Clustering VII

AGNES – Agglomerative Nesting [KR90] II

Lance-Williams update equation have the general form:

$$D(A \cup B, C) = \alpha_1 d(A, C) + \alpha_2 d(B, C) + \beta d(A, B) + \gamma |d(A, C) - d(B, C)|$$

Several (but not all) linkages can be expressed in this form (for distances):

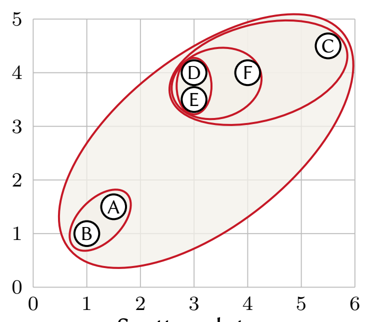
	α_1	α_2	β	γ
Single-linkage	$1/2$	$1/2$	0	$-1/2$
Complete-linkage	$1/2$	$1/2$	0	$+1/2$
Average-group-linkage (UPGMA)	$\frac{\|A\|}{\|A\|+\|B\|}$	$\frac{\|B\|}{\|A\|+\|B\|}$	0	0
McQuitty (WPGMA)	$1/2$	$1/2$	0	0
Centroid-linkage (UPGMC)	$\frac{\|A\|}{\|A\|+\|B\|}$	$\frac{\|B\|}{\|A\|+\|B\|}$	$\frac{-\|A\|\|B\|}{(\|A\|+\|B\|)^2}$	0
Median-linkage (WPGMC)	$1/2$	$1/2$	$-1/4$	0
Ward	$\frac{\|A\|+\|C\|}{\|A\|+\|B\|+\|C\|}$	$\frac{\|B\|+\|C\|}{\|A\|+\|B\|+\|C\|}$	$\frac{-\|C\|}{\|A\|+\|B\|+\|C\|}$	0

Hierarchical Agglomerative Clustering VIII

AGNES – Agglomerative Nesting [KR90] III

We may need to merge non-adjacent rows!

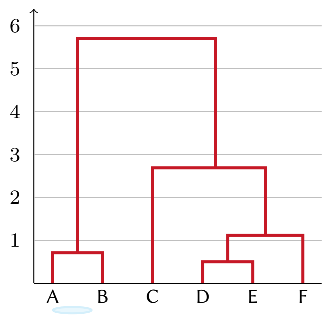
Example with complete linkage (= maximum of the distances):



Scatter plot

	A	B	C	D	E	F
A	0	0.71	5	2.92	2.5	3.54
B	0.71	0	5.70	3.61	3.20	4.24
C	5	5.70	0	2.55	2.69	1.58
D	2.92	3.61	2.55	0	0.5	1
E	2.5	3.20	2.69	0.5	0	1.12
F	3.54	4.24	1.58	1	1.12	0

Distance matrix



Dendrogram

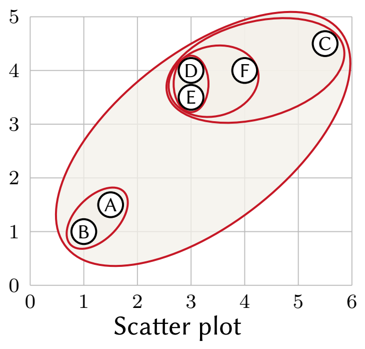
We don't know the optimum label positions in advance

Hierarchical Agglomerative Clustering VIII

AGNES – Agglomerative Nesting [KR90] III

In this very simple example, single and complete linkage are very similar

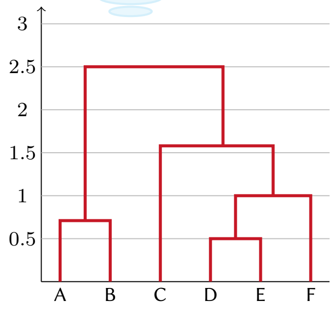
Example with single linkage (= minimum of the distances):



Scatter plot

	A	B	C	D	E	F
A	0	0.71	5	2.92	2.5	3.54
B	0.71	0	5.70	3.61	3.20	4.24
C	5	5.70	0	2.55	2.69	1.58
D	2.92	3.61	2.55	0	0.5	1
E	2.5	3.20	2.69	0.5	0	1.12
F	3.54	4.24	1.58	1	1.12	0

Distance matrix



Dendrogram

Hierarchical Agglomerative Clustering IX

Extracting clusters

At this point, we have the dendrogram – but not yet "clusters".

➡ E.g., set a distance limit, or stop at k clusters.

Complexity analysis:

1. Computing the distance matrix: $\mathcal{O}(n^2)$ time and memory.
2. Finding the maximum: $\mathcal{O}(n^2) \cdot i$
3. Updating the matrix: $\mathcal{O}(n) \cdot i$
4. Number of iterations: $i = \mathcal{O}(n)$

Total: $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ memory!

Better algorithms can run in "usually n^2" time [Sib73; And73; Def77].

➡ Hierarchical clustering does not scale to large data, code optimization matters [KSZ16].

Hierarchical Agglomerative Clustering X

Benefits and limitations

Benefits:

- Very general: any distance / similarity (for text: cosine!)
- Easy to understand and interpret
- Dendrogram visualization can be useful
- Many variants

Limitations:

- Scalability is the main problem (in particular, $\mathcal{O}(n^2)$ memory)
- Unbalanced cluster sizes (i.e., number of points)
- Outliers

k-means Clustering

Core concepts

The k-means problem:

- Divide data into k subsets (k is a parameter)
- Subsets represented by their *arithmetic mean* in each attribute $\mu_{C,d}$
- Optimize the *least squared* error $\text{SSQ} := \sum_C \sum_d \sum_{x_i \in C} (x_{i,d} - \mu_{C,d})^2$

History of least squares estimation (Legendre, Gauss):
`https://en.wikipedia.org/wiki/Least_squares#History`

- Squared errors put more weight on larger deviations
- Arithmetic mean is the maximum likelihood estimator of centrality
- Connected to the normal distribution

➡ k-means is a good choice, if we have k signals and normal distributed measurement error

k-means Clustering II

Sum of Squares objective

We can rearrange these sums because of communtativity

The sum-of-squares objective:

$$\text{SSQ} := \underbrace{\sum\nolimits_C}_{\text{every cluster}} \times \underbrace{\sum\nolimits_d}_{\text{every dimension}} \times \underbrace{\sum\nolimits_{x_i \in C}}_{\text{every point}} \underbrace{(x_{i,d} - \mu_{C,d})^2}_{\text{squared deviation from mean}}$$

For every cluster C and dimension d, the arithmetic mean minimizes

$$\sum\nolimits_{x_i \in C} (x_{i,d} - \mu_{C,d})^2 \quad \text{is minimized by} \quad \mu_{C,d} = \frac{1}{|C|} \sum\nolimits_{x_i \in C} x_{i,d}$$

For every point x_i, we can choose the cluster C to minimize SSQ, too.

Note: sum of squares $\equiv$ squared Euclidean distance:

$$\sum\nolimits_d (x_{i,d} - \mu_{C,d})^2 \equiv \|x_i - \mu_C\|^2 \equiv d^2_{\text{Euclidean}}(x_i, \mu_C)$$

We can therefore say that every point is assigned the "closest" cluster, *but* we cannot use arbitrary other distance functions in k-means (because the arithmetic mean only minimizes SSQ).

k-means Clustering III

The standard algorithm (Lloyd's algorithm)

(2.) and (3.) both minimize SSQ

The standard algorithm for k-means [Ste56; For65; Llo82]:

1. Choose k points randomly[4] as initial centers
2. Assign every point to the least-squares closest center
3. Update the centers with the arithmetic mean
4. Repeat (2.)-(3.) until no point is reassigned anymore

This is *not* the most efficient algorithm (despite everybody teaching this variant).
ELKI [Sch+15] contains $\approx$ 10 variants (e.g., Sort-Means [Phi02]; benchmarks in [KSZ16]).

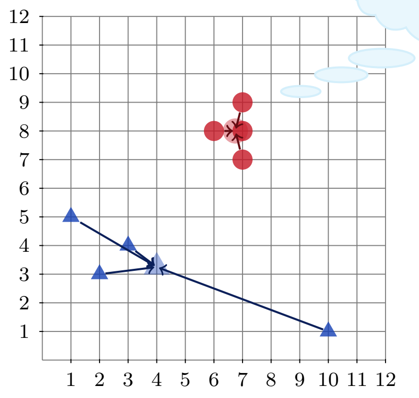
The name k-means was first used by MacQueen for a slightly different algorithm [Mac67].

k-means was invented several times, and has an interesting history [Boc07].

[4] or by any other rule, e.g., k-means++ [AV07] or predefined "seeds"

k-means Clustering IV

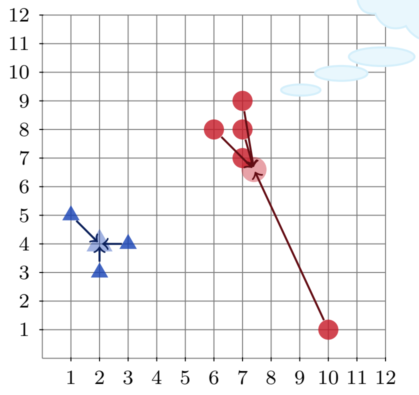
The standard algorithm (Lloyd's algorithm) II



k-means has converged
in the third iteration
with SSQ $= 61.5$

k-means Clustering V

The standard algorithm (Lloyd's algorithm) III

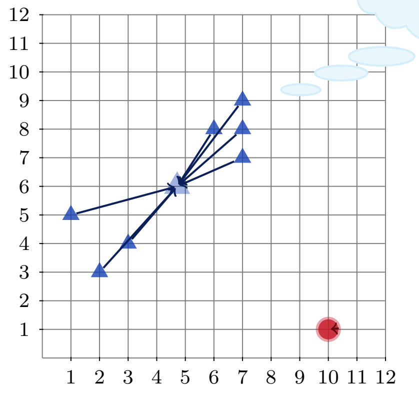


k-means has converged
in the second iteration
with SSQ $= 54.4$

Result with different starting centroids.

k-means Clustering VI

The standard algorithm (Lloyd's algorithm) IV



k-means has converged
in the second iteration
with SSQ $= 72.9$

Result with different starting centroids.

k-means Clustering VII

Non-determinism & non-optimality

Most k-means algorithms

- do not guarantee to find the *global optimum* (would be NP-hard – too expensive)
- give different local optima,[5] depending on the starting point

In practical use:

- data is never exact, or complete
- the "optimum"[6] result is not necessarily the most *useful*

➨ Usually, we gain little by finding the true optimum

➨ It is usually good enough to try a few random initializations and keep the "best"[6]

[5]In fact, the standard algorithm may fail to even find a local minimum [HW79].
[6]Least squares, i.e., lowest SSQ – this does not mean it will actually give the most insight.

k-means Clustering VIII

Complexity

In the standard algorithm:

1. Initialization is usually cheap, $O(k)$ (k-means++: $O(N \cdot k \cdot d)$ [AV07])
2. Reassignment is $O(N \cdot k \cdot d)$
3. Mean computation is $O(N \cdot d)$
4. Number of iterations $i \in 2^{\Omega(\sqrt{N})}$ [AV06] (but fortunately, usually $i \ll N$)
5. Total: $O(N \cdot k \cdot d \cdot i)$

Worst case is superpolynomial, but in practice the method will usually run much better than n^2.

We can force a limit on the number of iterations, e.g., $i = 100$, with little loss in quality usually.

In practice, often the fastest clustering algorithm we have / use.

k-means Clustering IX

k-means for text clustering

k-means cannot be used with arbitrary distances, but only with Bregman divergences [Ban+05].

Cosine similarity is closely connected to squared Euclidean distance (c.f. Assignment 2).

Spherical k-means [DM01] uses:

- Input data is normalized to have $\|x_i\| = 1$
- At each iteration, the new centers are normalized to $\mu'_C := \|\mu_C\| = 1$
- μ'_C minimizes average cosine similarity [DM01]

$$\sum_{x_i \in C} \langle x_i, \mu'_C \rangle \cong |C| - \sum_{x_i \in C} \|x_i, \mu'_C\|^2$$

- Sparse nearest-centroid computations in $O(d')$ where d' is the number of non-zero values
- Result is similar to a SVD factorization of the document-term-matrix [DM01]

k-means Clustering X

Choosing the "optimum" k for k-means

A key challenge of k-means is choosing k:

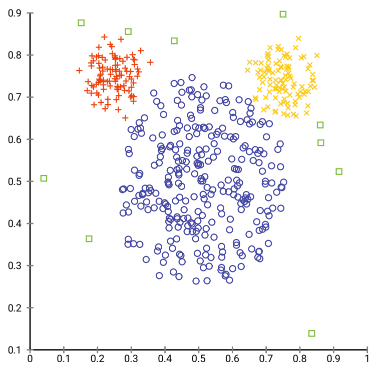
- Trivial to prove: $\mathrm{SSQ}_{\text{optimum},k} \geq \mathrm{SSQ}_{\text{optimum},k+1}$.
  - ➡ Avoid comparing SSQ for different k or different data (including normalization).
- $\mathrm{SSQ}_{k=N} = 0$ — "perfect" solution? No: useless.
- SSQ_k may exhibit an "elbow" or "knee": initially it improves fast, then much slower.
- Use alternate criteria such as Silhouette [Rou87], AIC [Aka77], BIC [Sch78; ZXF08].
  - ➡ Computing silhouette is $\mathcal{O}(n^2)$ – more expensive than k-means.
  - ➡ AIC, BIC try to reduce overfitting by penalizing model complexity (= high k).

  More details will come in evaluation section.
- Nevertheless, these measures are *heuristics* – other k can be better in practice!
- Methods such as X-means [PM00] split clusters as long as a quality criterion improves.

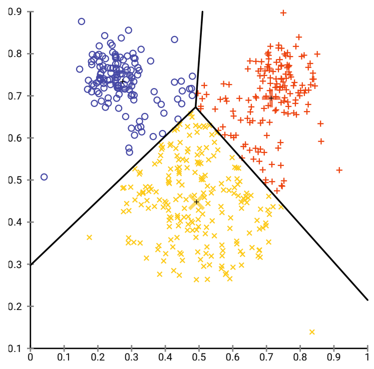
k-means Clustering XI

Choosing the "optimum" k for k-means II

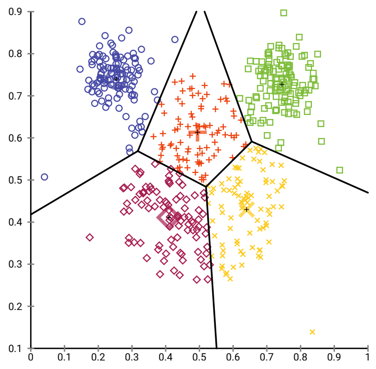
Toy "mouse" data set:



Best with $k = 3$:



Best with $k = 5$:

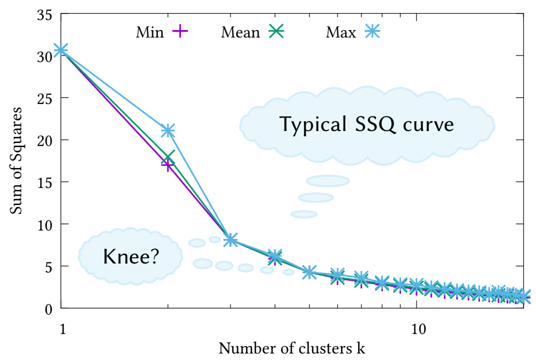


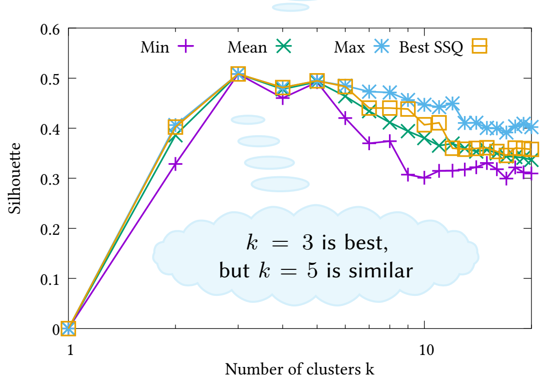
k-means Clustering XI

Choosing the "optimum" k for k-means II

0.5 is not considered to be a good Silhouette

Best results for 25 initializations, $k = 1 \ldots 20$:



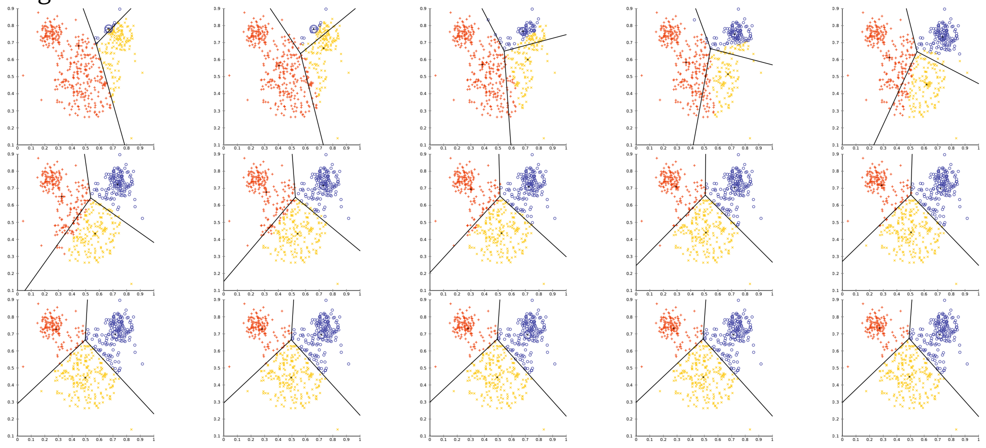


All tested measures either prefer 3 or 5 clusters.

k-means Clustering XII

Clusters changes are increasingly incremental

Convergence on Mouse data set with $k = 3$:



k-means Clustering XIII

k-means benefits and drawbacks

Benefits:

- Very fast algorithm ($O(k \cdot d \cdot N)$, if we limit the number of iterations)
- Convenient centroid vector for every cluster
  (We can analyze this vector to get a “topic”)
- Can be run multiple times to get different results

Limitations:

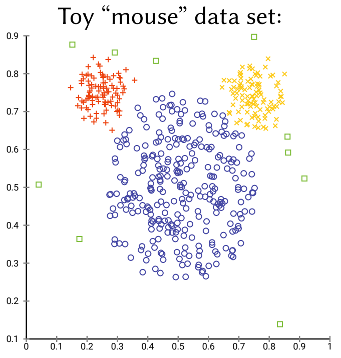
- Cannot be used with arbitrary distances
- Difficult to choose the number of clusters, k
- Does not produce the same result every time
- Sensitive to outliers (squared errors emphasize outliers)
- Cluster sizes can be quite unbalanced (e.g., one-element outlier clusters)

Expectation-Maximization Clustering

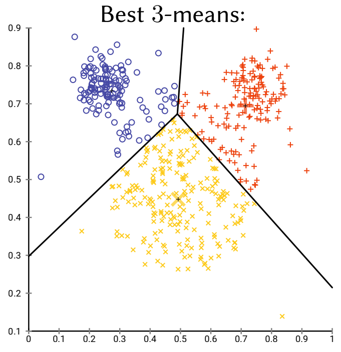
From k-means to Gaussian EM

k-means can not handle clusters with different “radius” well.

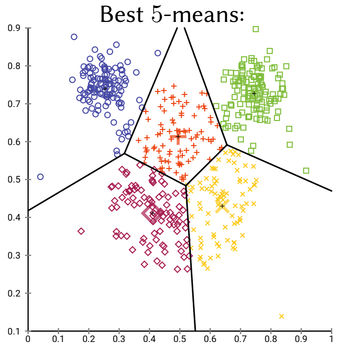
Toy “mouse” data set:



Best 3-means:



Best 5-means:



- could we estimate mean and radius?
- model the data with multivariate Gaussian distributions

Expectation-Maximization Clustering

Iterative refinement

EM (Expectation-Maximization) is the underlying principle in Lloyd's k-means:

1. Choose initial model parameters θ
2. Expect latent variables (e.g., cluster assignment) from θ and the data.
3. Update θ to maximize the likelihood of observing the data
4. Repeat (2.)-(3.) until a stopping condition holds

Recall Lloyd's k-means:

1. Choose k centers randomly (θ: random centers)
2. Expect cluster labels by choosing the nearest center as label
3. Update cluster centers with maximum-likelihood estimation of centrality
4. Repeat (2.)-(3.) until change = 0

Expectation-Maximization Clustering

Iterative refinement

EM (Expectation-Maximization) is the underlying principle in Lloyd's k-means:

1. Choose initial model parameters θ
2. Expect latent variables (e.g., cluster assignment) from θ and the data.
3. Update θ to maximize the likelihood of observing the data
4. Repeat (2.)-(3.) until a stopping condition holds

Gaussian Mixture Modeling (GMM): [DLR77]

1. Choose k centers randomly, and unity covariance ($\theta = (\mu_1, \Sigma_1, \mu_2, \Sigma_2, \ldots \mu_k, \Sigma_k)$)
2. Expect cluster labels based on Gaussian distribution density
3. Update Gaussians with mean and covariance matrix
4. Repeat (2.)-(3.) until change $< \epsilon$

Gaussian Mixture Modeling I

Expectation-Maximization

Expectation-step: For every point p, and cluster center μ_i with covariance matrix Σ_i compute:

$$\text{pdf}(p, \mu_i, \Sigma_i) := \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \cdot e^{-\frac{1}{2}\left((p-\mu_i)^T \Sigma_i^{-1} (p-\mu_i)\right)}$$

Estimate point weights (cluster membership)

$$w_{pi} := \frac{\text{pdf}(p, \mu_i, \Sigma_i)}{\sum_j \text{pdf}(p, \mu_j, \Sigma_j)}$$

w proportional to pdf
$w_{pi} \propto \text{pdf}(p, \mu_i, \Sigma_i)$

Maximization step: Use weighted mean and weighted covariance to recompute cluster model.

$$\mu_{i,x} = \frac{1}{\sum_p w_{pi}} \sum\nolimits_p w_{pi} p_x$$

$$\Sigma_{i,x,y} = \frac{1}{\sum_p w_{pi}} \sum\nolimits_p w_{pi}(p_x - \mu_x)(p_y - \mu_y)$$

weighted $\text{mean}(X)$ / $\text{cov}(X, Y)$

Gaussian Mixture Modeling II

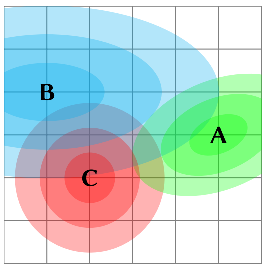
Fitting multiple Gaussian distributions to data

Probability density function of a multivariate Gaussian:

$$\text{pdf}(p, \mu, \Sigma) := \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \cdot e^{-\frac{1}{2}\left((p-\mu)^T \Sigma^{-1} (p-\mu)\right)}$$

If we constrain Σ we can control the cluster shape:

- We always want symmetric and positive semi-definite
- Σ covariance matrix: rotated ellipsoid (A)
- Σ diagonal (“variance matrix”): ellipsoid (B)
- Σ scaled unit matrix: spherical (C)
- Same Σ for all clusters, or different Σ_i each



Gaussian Mixture Modeling III

Understanding the Gaussian density

Multivariate normal distribution:

$$\text{pdf}(p, \mu, \Sigma) := \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \cdot e^{-\frac{1}{2}\left((p-\mu)^T \Sigma^{-1} (p-\mu)\right)}$$

1-dimensional normal distribution:

$$\text{pdf}(x, \mu, \sigma) := \frac{1}{\sqrt{(2\pi)\sigma^2}} \cdot e^{-\frac{1}{2}\left((x-\mu)\sigma^{-2} (x-\mu)\right)}$$

Normalization (to a total volume of 1) and squared deviation from center
Compare this to Mahalanobis distance:

$$d_{\mathit{Mahalanobis}}(x, \mu, \Sigma)^2 := (x-\mu)^T \Sigma^{-1} (x-\mu)$$

➡ Σ/Σ^{-1} plays a central role here, the remainder is squared Euclidean distance!

Gaussian Mixture Modeling IV

Inverse covariance matrix – Σ^{-1}

Covariance matrixes are symmetric, non-negative on the diagonal, and can be inverted.
(This may need a robust numerical implementation.)

We can decompose this using

$$V \Lambda V^{-1} = \Sigma \quad \equiv \quad V \Lambda^{-1} V^{-1} = \Sigma^{-1}$$

where V contains the eigenvectors and Λ contains the eigenvalues.

➡ Interpret this decomposition as $V \cong$ rotation, $\Lambda \cong$ squared scaling!

(Recall foundations: PCA and SVD)

Gaussian Mixture Modeling V

Inverse covariance matrix – Σ^{-1} II

Build Ω using $\omega_i = 1/\sqrt{\lambda_i} = \lambda_i^{-\frac{1}{2}}$. Then $\Omega\Omega = \Lambda^{-1}$, and $\Omega^T = \Omega$.

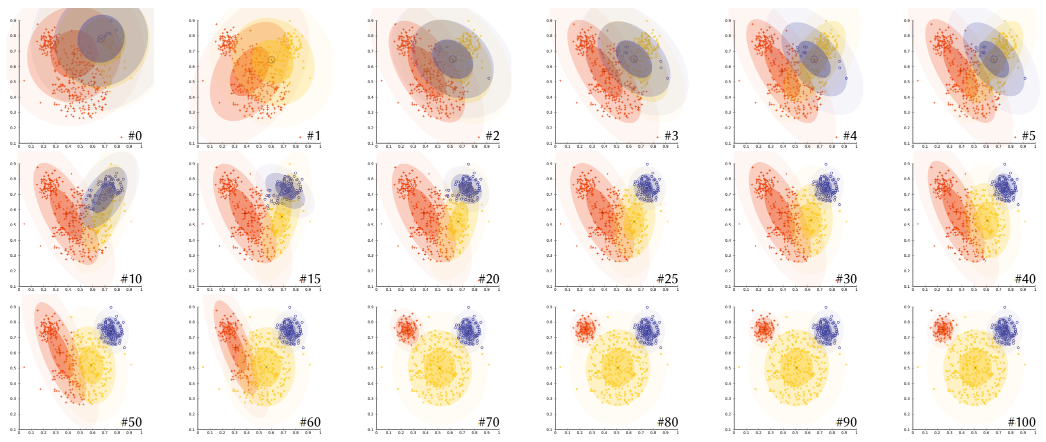
$$\begin{aligned}\Sigma^{-1} &= V\Lambda^{-1}V^{-1} = V\Omega^T\Omega V^T = (\Omega V^T)^T\Omega V^T \\ d^2_{\text{Mahalanobis}} &= (x-\mu)^T\Sigma^{-1}(\Omega V^T)^T\Omega V^T(x-\mu) \\ &= \langle \Omega V^T(x-\mu), \Omega V^T(x-\mu)\rangle \\ &= \left\|\Omega V^T(x-\mu)\right\|^2\end{aligned}$$

➨ Mahalanobis ≈ Euclidean distance after PCA

Gaussian Mixture Modeling VII

Soft-assignment changes slower than k-means

Clustering mouse data set with $k = 3$:



Expectation-Maximization Clustering

Clustering text data

We cannot use Gaussian EM on text:

- Text is not Gaussian distributed.
- Text is discrete and sparse, Gaussians are continuous.
- Covariance matrixes have $\mathcal{O}(d^2)$ entries:
  - Memory requirements (text has a very high dimensionality d)
  - Data requirements (to reliably estimate the parameters, we need very many data points)
  - Matrix inversion is even $\mathcal{O}(d^3)$

But the general EM principle can be used with *other distributions*.

For example: mixture of Bernoulli or multinomial distributions

Mixture of Bernoulli Distributions

EM Clustering meets Bernoulli Naïve Bayes [MRS08]

The Bernoulli model uses boolean vectors, indicating the presence of terms.

A cluster i is modeled a weight α_i and term frequencies $q_{i,t}$.

Multivariate Bernoulli probability of a document x in cluster i:

$$P(x \mid i, q_i) = \left(\prod\nolimits_{t \in x} q_{i,t}\right) \left(\prod\nolimits_{t \notin x} 1 - q_{i,t}\right)$$

Mixture of clusters $1 \dots k$ with weights $\alpha_1 \dots \alpha_k$:

$$P(x \mid \alpha, q) = \sum\nolimits_{i=1}^{k} \alpha_i \left(\prod\nolimits_{t \in x} \beta_{i,t}\right) \left(\prod\nolimits_{t \notin x} 1 - \beta_{i,t}\right)$$

Note: when implementing this, we need a Laplacian correction to avoid zero values!

Mixture of Bernoulli Distributions

Probabilistic generative model

This equation arises, if we *assume* the data is generated by:

1. Choose a cluster i with probability α_i
2. For every token t, include it in the document with probability $\beta_{i,t}$

This is a *very* simple model:

- No word frequency
- No word order
- No word correlations

If we would use this to really generate documents, they would be gibberish.

But naïve Bayes classification uses this, too, and it often works!

Mixture of Bernoulli Distributions

Expectation-Maximziation algorithm

To learn the weights α, β we can employ EM:

1. Choose $\alpha_i = 1/k$, and choose k documents as initial β_i (similar to k-means).
2. Expectation step:

$$P(x \in C_i \mid \alpha, \beta) = \frac{\alpha_i \left(\prod_{t \in x} \beta_{i,t}\right) \left(\prod_{t \notin x} 1 - \beta_{i,t}\right)}{\sum_{j=1}^{k} \alpha_j \left(\prod_{t \in x} \beta_{j,t}\right) \left(\prod_{t \notin x} 1 - \beta_{j,t}\right)}$$

3. Maximization step:

$$\beta_{i,t} = \frac{\sum_x P(x \in C_i \mid \alpha, \beta) \mathbb{1}(t \in x)}{\sum_x P(x \in C_i \mid \alpha, \beta)}$$

$$\alpha_i = \frac{\sum_x P(x \in C_i \mid \alpha, \beta)}{N}$$

4. Repeat (2.)-(3.) until change $< \epsilon$.

Probability of a document in cluster i containing token t $\mathbb{1}(c) = 1$ if c true else 0

What share of all documents is in the cluster?

Mixture of Bernoulli Distributions

Expectation-Maximziation algorithm

To learn the weights α, β we can employ EM:

1. Choose $\alpha_i = 1/k$, and choose k documents as initial β_i (similar to k-means).
2. Expectation step:

$$P(x \in C_i \mid \alpha, \beta) \propto \alpha_i \left(\prod\nolimits_{t \in x} \beta_{i,t}\right) \left(\prod\nolimits_{t \notin x} 1 - \beta_{i,t}\right)$$

3. Maximization step:

$$\beta_{i,t} = \frac{\sum_x P(x \in C_i \mid \alpha, \beta) \mathbb{1}(t \in x)}{\sum_x P(x \in C_i \mid \alpha, \beta)}$$

$$\alpha_i \propto \sum\nolimits_x P(x \in C_i \mid \alpha, \beta)$$

4. Repeat (2.)-(3.) until change $< \epsilon$.

Probability of a document in cluster i containing token t $\mathbb{1}(c) = 1$ if c true else 0

What share of all documents is in the cluster?

Mixture of Bernoulli Distributions

From Bernoulli to multinomial

The Bernoulli model is simple, but it ignores quantitative information completely.

The closest distribution that uses quantity is the multinomial distribution.

Again, this is similar to multinomial naïve Bayes classification.

Mixture of Multinomial Distributions

Clustering text data

The multinomial mixture model:

$$P(x \mid \alpha, \beta) := \sum_{j=1}^{k} \alpha_j \frac{\text{len}(x)!}{\prod_{t=1}^{d} \text{tf}_{t,x}!} \prod_{t=1}^{d} \beta_{t,j}^{\text{tf}_{t,x}}$$

$\sum^k$	sum of multinomial distributions ("clusters", or "topics", index $j = 1 \ldots k$)
α_j	relative size of topic j (α is a vector of length k)
$\frac{\text{len!}}{\prod \text{tf!}}$	number of permutations of the document with the same word vector
$\prod \beta^{\text{tf}}$	probability of seeing this word vector in topic j (β is a $k \times d$ matrix)
$\beta_{t,j}$	frequency of word t in topic j
tf	number of times the term occurred in the document (document-term-matrix)

Mixture of Multinomial Distributions

Probabilistic generative model

The model assumes our data set was generated by a process like this:

1. Sample a topic distribution α
2. For every topic t, sample a word distribution β_t
3. For every document d
   3.1 Sample a topic t from α
   3.2 Sample l words from the word distribution β_t

Note: this is *not* what we *do*, but our underlying assumptions.

Mixture of Multinomial Distributions

Probabilistic generative model II

We want to apply Bayes' rule:

$$P(\alpha, \beta \mid X) \propto P(X \mid \alpha, \beta)\, P(\alpha) P(\beta)$$

Because α and β are independent, and $P(X)$ can be treated as constant.

$$P(\alpha, \beta \mid X) \propto \left(\prod_{x \in X} \sum_{j=1}^{k} \alpha_j \prod_{t=1}^{d} \beta_{t,j}^{\mathrm{tf}_{t,x}} \right) \prod_{j=1}^{k} \alpha_j^{\lambda_\alpha - 1} \prod_{j=1}^{k} \prod_{t=1}^{d} \beta_{t,j}^{\lambda_\beta - 1}$$

By disregarding everything that does not depend on α, β.

Unfortunately, maximizing this directly is in general intractable.

Mixture of Multinomial Distributions

Back to Expectation Maximization

Expectation step:

$$P(x \in j \mid \alpha, \beta) \propto \alpha_j \prod\nolimits_{t=1}^{d} \beta_{t,j}^{\mathrm{tf}_{t,x}}$$

by removing shared terms independent of j

Maximization step:

$$\alpha_j \propto \lambda_\alpha - 1 + \sum_{x \in X} P(x \in j \mid \alpha, \beta)$$

$$\beta_{t,j} \propto \lambda_\beta - 1 + \sum_{x \in X} \mathrm{tf}_{t,x}\, P(x \in j \mid \alpha, \beta)$$

This is called PLSA/PLSI [Hof99], and will be discussed in more detail in the next chapter.

Mixture of Multinomial Distributions

Results with Expectation Maximization

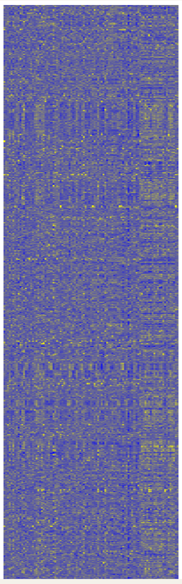
Results reported with this approach are mixed:

- Sensitive to initialization [MRS08; MS01]
  because there are many local optima. E.g., use k-means result as starting point.
- Sensitive to rare words
- Converges fast to binary assignments, usually
  (not necessarily good, tends to get stuck in local optima because of this)

Many more algorithms use this general EM optimization procedure!
E.g., for clustering web site navigation patterns (clickstreams) [YH02; Cad+03; JZM04]

Biclustering & Subspace Clustering

Clustering attributes and variables



Popular in gene expression analysis.

- Every row is a gene
- Every column is a sample

$\Big\}$ or transposed

- Only a few genes are relevant
- No semantic ordering of rows or columns
- Some samples may be contaminated
- Numerical value may be unreliable, only "high" or "low"

➡ Key idea of biclustering: [CC00]
Find a subset of rows and columns (submatrix, after permutation),
such that all values are high/low or exhibit some pattern.

Biclustering

Bicluster patterns [CC00]

Some examples for bicluster patterns:

1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1

constant

1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6

rows

1	1	1	1	1	1
2	2	2	2	2	2
3	3	3	3	3	3
4	4	4	4	4	4
5	5	5	5	5	5
6	6	6	6	6	6

columns

1	2	3	4	5	6
2	3	4	5	6	7
3	4	5	6	7	8
4	5	6	7	8	9
5	6	7	8	9	10
6	7	8	9	10	11

additive

1	2	4	0	4	3
3	6	12	0	12	9
2	4	8	0	8	6
4	8	16	0	16	12
1.5	3	6	0	6	4.5
0.5	1	2	0	2	1.5

multiplicative

Clusters *may* overlap in rows and columns!

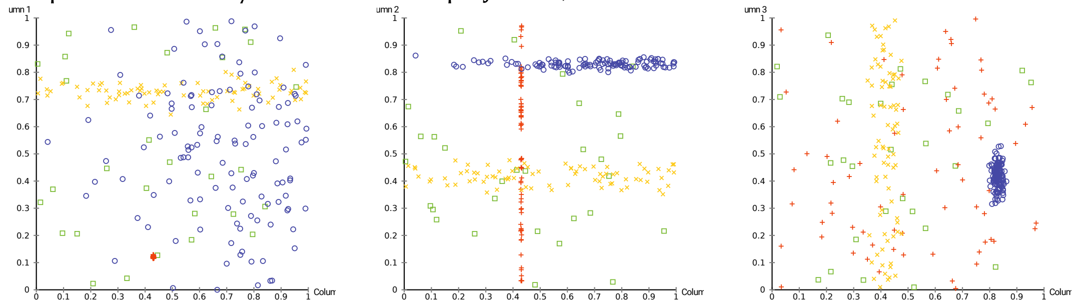
Patterns will never be this ideal, but noisy!

Many algorithms focus on the constant pattern type only, as there are $\mathcal{O}(2^{N \cdot d})$ possibilities.

Subspace Clustering

Density-based clusters in subspaces

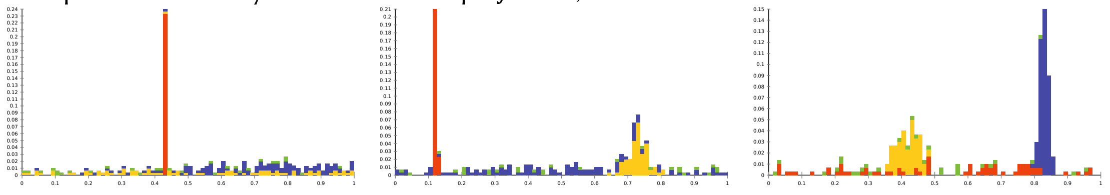
Subspace clusters may be visible in one projection, but not in another:



Subspace Clustering

Density-based clusters in subspaces

Subspace clusters may be visible in one projection, but not in another:



Popular key idea:

- Find dense areas in 1-dimensional projections
- Combine subspaces as long as the cluster remains dense

Examples: CLIQUE [Agr+98], PROCLUS [Agg+99], SUBCLU [KKK04]
There also exist “correlation clustering”, for rotated subspaces.

Biclustering & Subspace Clustering

Relationship to text clustering and topic modeling

While these algorithms have similar ideas to text clustering,
there are some subtle differences that make them not work well with text:

- Text is sparse, and we need to treat 0 specially
  (We do not want to find a cluster “all attributes are zero, except for a few”)
- We do not see “dense” regions of non-zero values
- We do not have the biological notion of “highly expressed” for genes
  What is a high value in TF-IDF? Everything except 0?

However, on some special (non-natural) text, these methods *may* work, e.g.:

- Tags / Keywords
- Log files

Frequent Itemset Mining

Finding co-occurrences

Market basket analysis:

- Which items are frequently bought together?
- If a customer has bought A and B, should I offer C?
- Cross-marketing opportunities?
- Optimize product arrangement in the store?

➡ For more details on the market basket analysis scenario, see KDD lecture!

Frequent Itemset Mining

Transaction data

Data model of frequent itemset mining:

- *Items* $I = \{i_1, \ldots, i_K\}$: literals, e.g., products or product groups
- *Itemset*: a set $X \subseteq I$ of items
- k*-itemset*: itemset of *length* $|X| = k$
- *Transactions* $T = \{X_1, \ldots, X_N\}$, $X \subseteq I$: observed itemsets and stored in the *Database*
- *Support* of an itemset X: $|\{X_i \in T \mid X \subseteq X_i\}|$

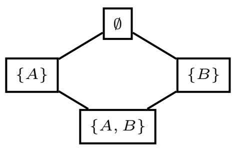
Notes:

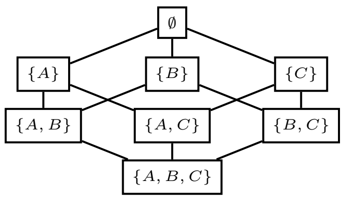
- itemsets are usually kept sorted for efficiency
- items are usually abstracted to product types, e.g., Milk, rather than brands and package sizes
- quantity information is usually not used

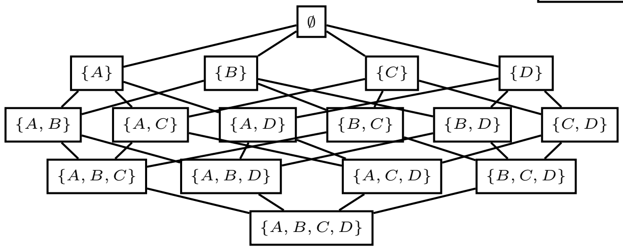
Frequent Itemset Mining

Combinatorial explosion

Too many possibilities (2^K):



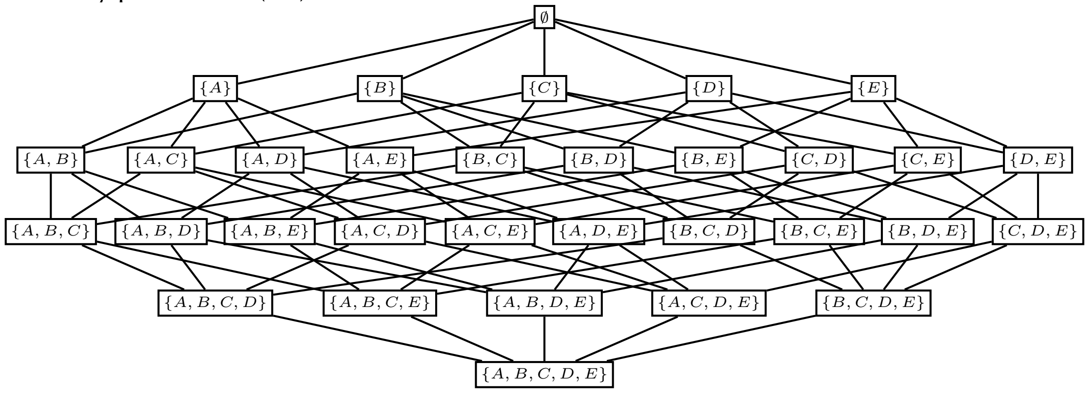




Frequent Itemset Mining

Combinatorial explosion

Too many possibilities (2^K):



Frequent Itemset Mining

APRIORI [AS94]

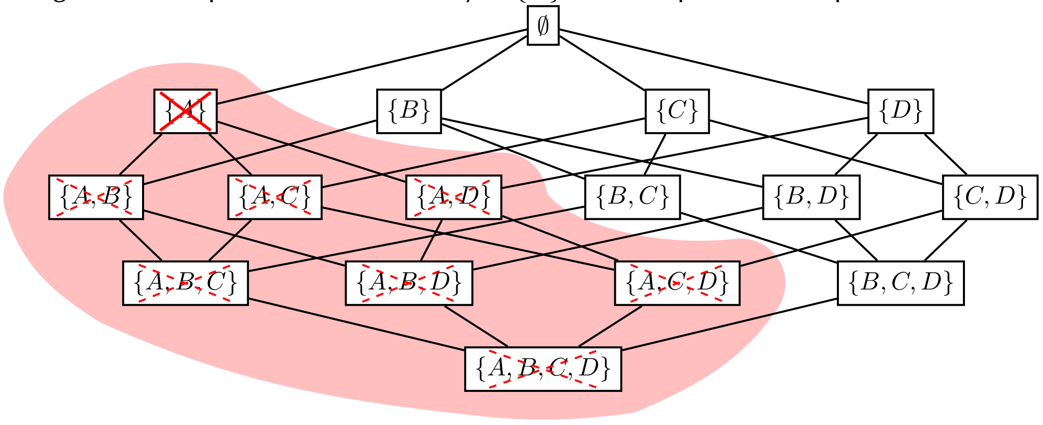
Incremental construction (with a minimum support threshold):

1. Find frequent 1-itemsets by counting.
2. Given the k-itemsets, find the $k+1$-itemsets.
   For this, generate as few candidates as possible using *monotonicity*:
   - Support is monotone decreasing: $\text{Support}(X \cup \{i\}) \leq \text{Support}(X)$
   - Therefore, $\text{Support}(X = \{x_1, \ldots, x_{k+1}\}) \leq \min_i \text{Support}(X \setminus \{x_i\})$ and $X \setminus \{x_i\} \notin k$-itemsets $\Rightarrow \text{Support}(X \setminus \{x_i\}) <$ minSupp $\Rightarrow \text{Support}(X) <$ minSupp
   - Optimization: only combine itemsets which agree on the first $k-1$ items: $\{x_1, \ldots, x_{k-1}, x_k\} \cup \{x_1, \ldots, x_{k-1}, \underline{x'_k}\} = \{x_1, \ldots, x_{k-1}, x_k, \underline{x'_k}\}$
   - Optimization: keep everything sorted, then $\{x_1, \ldots, x_{k-1}, \_\}$ are sequential
   - Check if all other subsets $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k\} \in k$-itemsets for $i = 1 \ldots k-2$
3. Use *a single scan* of the database for each k, count support of candidates
4. Discard candidates with too little support
5. Stop if no more itemsets can be found in the next round ($|k+1\text{-itemsets}| < k+2$)

Frequent Itemset Mining

Combinatorial explosion

Pruning the search space with monotonicity: if $\{A\}$ is not frequent, we can prune



Frequent Itemset Mining Example

NetFlix data

2 GB of movie ratings from NetFlix, ca. 2006.
Simplify: only 5 star ratings $\Rightarrow$ 23 million items in 480189 transactions (users)
But: even with minimum support 1000, we get billions of frequent itemsets!

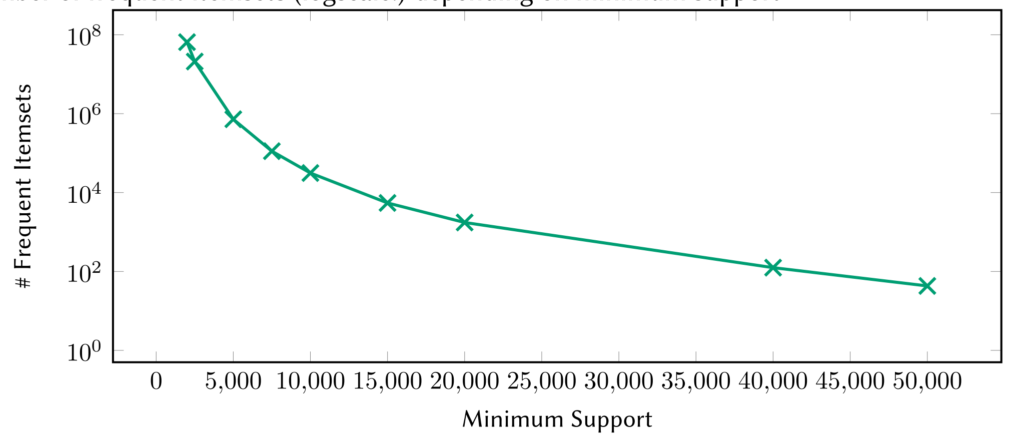
Maximum supported k-itemsets:

$k = 1$	96535	Lord of the Rings: The Two Towers
$k = 2$	77878	above + Lord of the Rings: The Fellowship of the Ring
$k = 3$	66083	above + Lord of the Rings: The Return of the King
$k = 4$	43177	above + Lord of the Rings: The Two Towers: Extended Edition
$k = 5$	40123	above + Lord of the Rings: The Fellowship of the Ring: Extended Edition
$k = 6$	36267	above + Lord of the Rings: The Return of the King: Extended Edition
$k = 7$	22429	above + Star Wars: Episode V: The Empire Strikes Back
$k = 8$	18789	above + Star Wars: Episode IV: A New Hope
$k = 9$	16497	above + Star Wars: Episode VI: Return of the Jedi
$k = 10$	12241	above + Raiders of the Lost Ark

Frequent Itemset Mining

Combinatorial explosion – Example

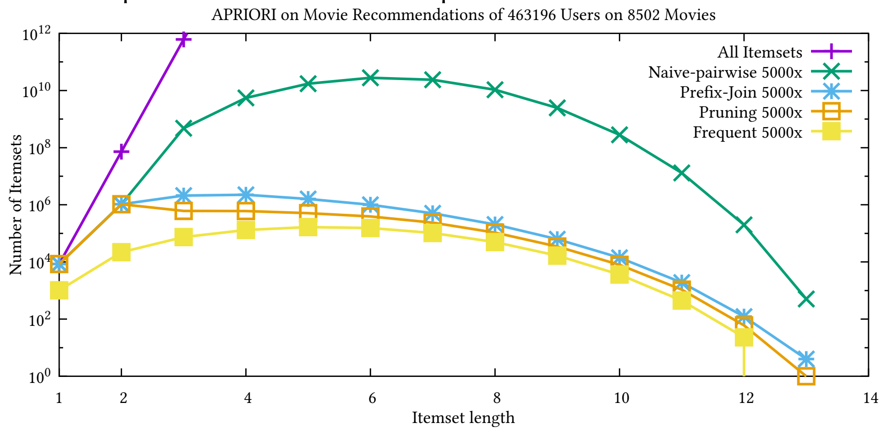
Number of frequent itemsets (logscale!) depending on minimum support:



Frequent Itemset Mining

Combinatorial explosion – Example

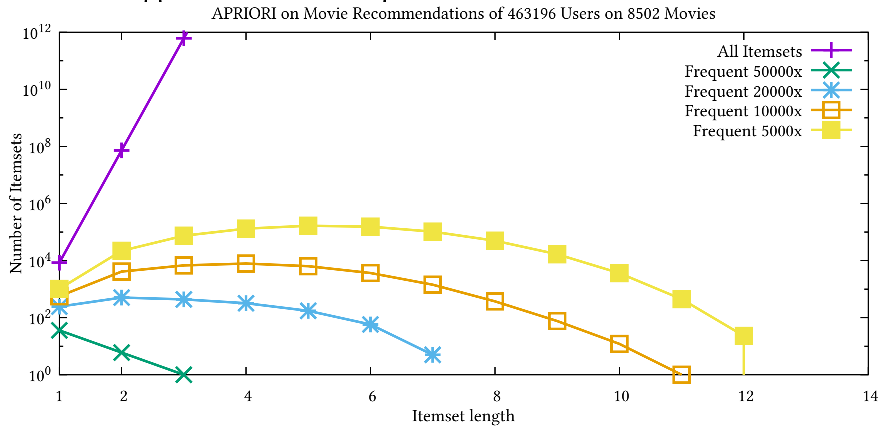
Effect of APRIORI optimizations on the search space:



Frequent Itemset Mining

Combinatorial explosion – Example

Effect of minimum support on the search space:



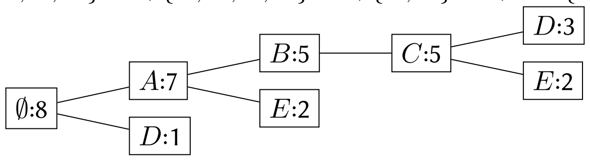
Frequent Itemset Mining

FP-Growth [HPY00]

Key-ideas of FP-Growth:

- many itemsets are duplicate or similar
- a prefix-tree-like aggregation can exploit redundancy
- the tree can often be held in main memory even when the database cannot
- find frequent patterns from the aggregated tree via projection (see KDD lecture)

E.g., transactions $\{A, B, C, D\} \times 3$, $\{A, B, C, E\} \times 2$, $\{A, E\} \times 2$, and $\{D\} \times 1$:



The FP-tree is a *compressed summary* of the transaction database.

Frequent Itemset Mining

Text and transactions

Comparison to text:

- item $\sim$ token (word)
- itemset $\sim$ binary term vector
- transactions $\sim$ documents

➡ Use frequent itemset mining to find patterns in text?

Does not work well on natural text: common words are frequent, interesting words are rare.
$\Rightarrow$ frequent itemsets involve mostly frequent words

On non-standard text such as tags, log files, etc. this *can* work well!

Evaluation of Clustering

Different kinds of evaluation

We can distinguish between four kinds of evaluation:

- Unsupervised evaluation (usually based on distance / deviation)
  Statistics such as SSQ, Silhouette, Davies-Bouldin, …
- Supervised evaluation based on labels
  Indexes such as Adjusted Rand Index, Purity, …
- Indirect evaluation
  Based on the performance of some other (usually supervised) algorithm in a later stage
  E.g., how much does classification improve, if we use the clustering as feature
- Expert evaluation
  Manual evaluation by a domain expert

Unsupervised Evaluation of Clusterings

Underlying principles

Recall the basic idea of distance-based clustering algorithms:

- Items in the same cluster should be more similar
- Items in other clusters should be less similar

➡ compare the distance within the same cluster to distances to other clusters

Some simple approaches:

$$\text{MeanDistance}(C) := \tfrac{1}{N}\sum\nolimits_{C_i}\sum\nolimits_{x\in C_i} d(x,\mu_{C_i})$$

$$\text{MeanSquaredDistance}(C) := \tfrac{1}{N}\sum\nolimits_{C_i}\sum\nolimits_{x\in C_i} d^2(x,\mu_{C_i})$$

$$\text{RMSD}(C) := \sqrt{\tfrac{1}{N}\sum\nolimits_{C_i}\sum\nolimits_{x\in C_i} d^2(x,\mu_{C_i})}$$

Variance-based Evaluation

Explained Variance I

Variance is proportional to the pairwise deviations:

$$\begin{aligned}
\operatorname{Var} X = E[X^2] - E[X]^2 &= \tfrac{1}{N}\sum\nolimits_x x^2 - \left(\tfrac{1}{N}\sum\nolimits_x x\right)^2 \\
&= \tfrac{1}{N}\sum\nolimits_x x^2 - \tfrac{1}{N}\sum\nolimits_x x \cdot \tfrac{1}{N}\sum\nolimits_y y \\
&= \tfrac{1}{N}\sum\nolimits_x x^2 - \tfrac{1}{N^2}\sum\nolimits_{x,y} xy \\
&= \tfrac{1}{2N^2}\left(N\sum\nolimits_x x^2 + N\sum\nolimits_y y^2 - 2\sum\nolimits_{x,y} xy\right) \\
&= \tfrac{1}{2N^2}\sum\nolimits_{x,y}\left(x^2 - 2xy + y^2\right) \\
&= \tfrac{1}{2N^2}\sum\nolimits_{x,y}(x-y)^2
\end{aligned}$$

➡ $2N^2 \operatorname{Var} X = \sum\nolimits_{x,y}(x-y)^2$

Variance-based Evaluation

Explained Variance II

We can decompose the variance for clusters $C_1, \dots, C_k$:

$$2N^2 \operatorname{Var} X = \sum\nolimits_{x,y} (x-y)^2$$

$$= \underbrace{\sum\nolimits_x \sum\nolimits_y (x-y)^2}_{\text{Total Sum of Squares (TSS)}} = \underbrace{\sum\nolimits_{C_i} \underbrace{\sum\nolimits_{x,y \in C_i} (x-y)^2}_{\text{within cluster } C_i}}_{\text{Within Cluster Sum of Squares (WCSS)}} + \underbrace{\sum\nolimits_{C_i \neq C_j} \underbrace{\sum\nolimits_{x \in C_i, y \in C_j} (x-y)^2}_{\text{between clusters } C_i, C_j}}_{\text{Between Cluster Sum of Squares (BCSS)}}$$

➨ Because total sum of squares TSS is constant, minimizing WCSS = maximizing BCSS

Explained variance := BCSS / TSS = (TSS − WCSS) / TSS $\in [0, 1]$

Note: k-means tries to find a local optimum of WCSS.

Note: Ward-linkage joins clusters with least increase in WCSS.

Silhouette

Distance to second nearest cluster [Rou87]

Define the mean distance of x to its own cluster, and to the closest other cluster:

$$\begin{aligned} a(x \in C_i) &:= \phantom{\min_{C_j \neq C_i}} \operatorname{mean}_{y \in C_i, y \neq x} d(x,y) \\ b(x \in C_i) &:= \min\nolimits_{C_j \neq C_i} \operatorname{mean}_{y \in C_j} d(x,y) \end{aligned}$$

The silhouette width of a point x (can be used for plotting) then is:

$$s(x) := \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}$$

The silhouette of a clustering C then is:

$$\text{Silhouette}(C) := \operatorname{mean}_x s(x)$$

Silhouette: 1 if $a \ll b$, 0 if $a = b$, and -1 if $a \gg b$.
An average Silhouette of > 0.5 is considered “reasonable”, > 0.7 is “strong”.

Silhouette

Challenges with Silhouette

- The complexity of Silhouette is: $\mathcal{O}(n^2)$
  $\Rightarrow$ does not scale to large data sets.
  Simplified Silhouette: use distances to cluster centers instead of average distances, $\mathcal{O}(n \cdot k)$.
- When a cluster has a single point, $a(x)$ is not defined.
  Rousseeuw [Rou87] suggests to use $s(x) = 0$ then.
- Which distance should we use, e.g., with k-means – Euclidean, or squared Euclidean?
- In high-dimensional data, $a(x) \to b(x)$ due to the curse of dimensionality. Then $s(x) \to 0$

Davies-Bouldin Index

Scatter vs. separation [DB79]

Power mean (with power p) of $(\|x-y\|_p)^p = \sum_i (x_i - y_i)^p$

Let the scatter of a cluster C_i be (recall L_p-norms):

$$S_i := \left(\tfrac{1}{|C_i|} \sum_{x \in C_i} \|x - \mu_{C_i}\|_p^p\right)^{1/p}$$

Let the separation of clusters be:

$$M_{ij} := \left\|\mu_{C_i} - \mu_{C_j}\right\|_p$$

For $p=2$, S_i is standard deviation, M_{ij} is the Euclidean distance!

The similarity of two clusters then is defined as:

$$R_{ij} := \frac{S_i + S_j}{M_{ij}}$$

Clustering quality is the average maximum similarity:

$$\mathrm{DB}(C) := \mathrm{mean}_{C_i}\ \max_{C_j \neq C_i}\ R_{ij}$$

A small Davies-Bouldin index is better, i.e., $S_i + S_j \ll M_{ij}$, scatter $\ll$ distance

Other Internal Clustering Indexes

Many more indexes

There have been many more indexes proposed over time:

- Dunn index: cluster distance / maximum cluster diameter [Dun73; Dun74]
- Calinski-Harabasz variance ratio criterion [CH74]:

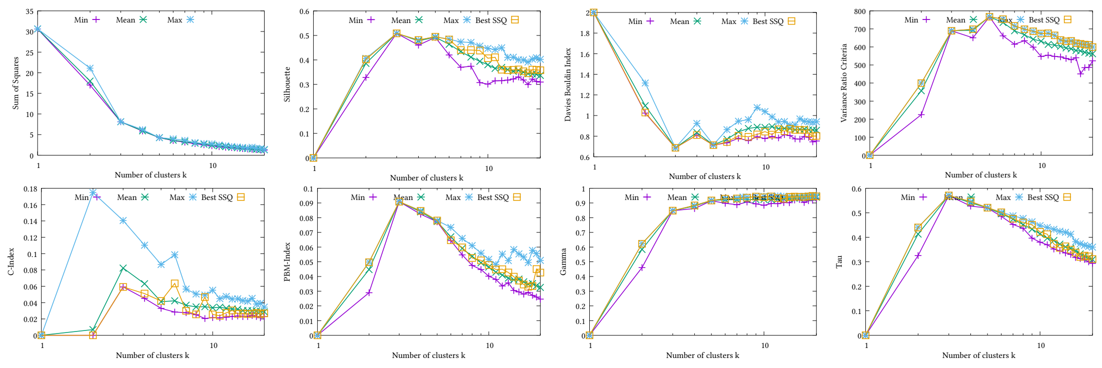
$$\frac{\mathrm{BCSS}/(k-1)}{\mathrm{WCSS}/(N-k)} = \frac{N-k}{k-1}\,\frac{\mathrm{BCSS}}{\mathrm{WCSS}}$$

- Gamma and Tau: P(within-cluster distance $<$ between-cluster distance) [BH75]
- C-Index: sum of within-cluster distances / same number of smallest distances [HL76]
- PBM Index: distance to cluster center / distance to total center [PBM04]
- DBCV: Density based cluster validation [Mou+14]
- …

Unsupervised Cluster Evaluation

Examples: Mouse data

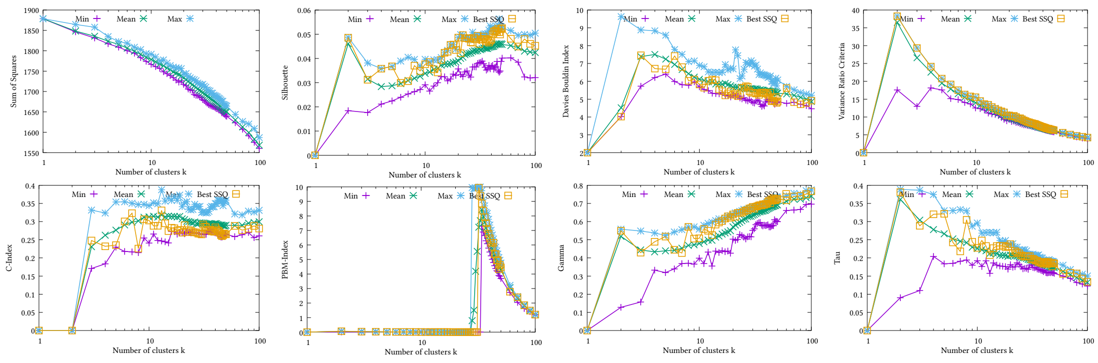
Revisiting k-means on the mouse data:



Unsupervised Cluster Evaluation

Examples: Tutorial's Wikipedia data set

Revisiting k-means on the Wikipedia data set (c.f., tutorials):



Supervised Cluster Evaluation

Evaluation with a "ground truth"

External evaluation measures assume we know true clusters.

In the following, every point has a cluster $C(x)$ and a true class $K(x)$.
The "raw data" (e.g., vectors) will *not* be used.

Popular in literature to compare algorithms.
Often, classification data is used, and it is assumed that good clusters = the classes.

Often not usable with real data – no labels available.
Sometimes, we can at least label some data, or treat some properties as potential labels, then choose the clustering that makes "most sense".

Supervised Cluster Evaluation

Evaluation with a "ground truth" II

The matching problem:

- Clusters C are usually enumerated $1, 2, 3, \ldots, k$
- True classes K are usually labeled with meaningful classes
- Which C is which class K?
  - ➨ Clustering is *not* classification, we cannot evaluate it the same way
- What if there are more clusters than classes?
- What if a cluster contains two classes?
- What if a class contains two clusters?

To overcome this

- Choose the best (C, K) matching with, e.g., the Hungarian algorithm (uncommon)
- Compare every cluster C to every class K

Supervised Cluster Evaluation

Purity, Precision and Recall

A simple measure popular in text clustering (but not much in "other" clustering):

$$\text{Purity}(C_i, K) := \max_{K_j} \frac{|C_i \cap K_j|}{|C_i|}$$

$$\text{Purity}(C, K) := \frac{1}{N} \sum_i |C_i| \text{Purity}(C_i, K) = \frac{1}{N} \sum_i \max_{K_j} |C_i \cap K_j|$$

⇒ A cluster, which *only* contains elements of class K_j has purity 1

➨ similar to "precision" in classification

But: every document is its own cluster has purity 1, is this really optimal?

We could also define a "recall" equivalent as $\max_{K_i} |C_j \cap K_i| / |K_i|$.
It works even worse: everything in a single cluster is optimal.

Supervised Cluster Evaluation

Pair-counting: classifying as related

Many cluster evaluation measures are based on pair-counting.

If (and only if) i and j are in the same cluster, then (i, j) is a pair.

This gives us a *binary, classification-like* problem:

		$C(i) = C(j)$: pair	$C(i) \neq C(j)$: no pair
$K(i) = K(j)$	pair	true positive (a)	false negative (c)
$K(i) \neq K(j)$	no pair	false positive (b)	true negative (d)

Objects are a pair if they are related
⇒ we are predicting which objects are related, and which are not
subject to the transitivity constraint: $(i, j) \wedge (j, k) \Rightarrow (i, k)$

Supervised Cluster Evaluation

Pair-counting measures

		$C(i) = C(j)$: pair	$C(i) \neq C(j)$: no pair
$K(i) = K(j)$	pair	true positive (a)	false negative (c)
$K(i) \neq K(j)$	no pair	false positive (b)	true negative (d)

$$\text{Precision} = \frac{a}{a+b} \qquad \text{Recall} = \frac{a}{a+c}$$

$$\text{Rand index [Ran71]} = \frac{a+d}{a+b+c+d} = \text{Accuracy}$$

$$\text{Fowlkes-Mallows [FM83]} = \sqrt{\text{Precision} \cdot \text{Recall}} = a \big/ \sqrt{(a+b) \cdot (a+c)}$$

$$\text{Jaccard} = \frac{a}{a+b+c}$$

$$\text{F}_1\text{-Measure} = \frac{2 \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2a}{2a+b+c}$$

$$\text{F}_\beta\text{-Measure} = \frac{(\beta^2+1) \cdot \text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}$$

$$\text{Adjusted Rand Index} = \frac{\text{Rand index} - E[\text{Rand index}]}{\text{Optimal Rand index} - E[\text{Rand index}]}$$

Supervised Cluster Evaluation

Mutual Information: Information-theoretic Evaluation [Mei03; Mei05; Mei12]

Mutual Information:

$$I(C,K) = \sum\nolimits_i \sum\nolimits_j P(C_i \cap K_j) \log \frac{P(C_i \cap K_j)}{P(C_i) \cdot P(K_j)}$$

$$= \sum\nolimits_i \sum\nolimits_j \frac{|C_i \cap K_j|}{N} \log \frac{N|C_i \cap K_j|}{|C_i| \cdot |K_j|}$$

Using $\log \frac{a}{b} = -\log \frac{b}{a}$

Entropy (c.f. preliminaries)

$$H(C) = -\sum\nolimits_i P(C_i) \log P(C_i) = -\sum\nolimits_i \frac{|C_i|}{N} \log \frac{|C_i|}{N} = I(C,C)$$

Normalized Mutual Information (NMI):[7]

$$\text{NMI}(C,K) = \frac{I(C,K)}{(H(C)+H(K))/2} = \frac{I(C,K)+I(K,C)}{I(C,C)+I(K,K)}$$

[7]There exist at least 5 variations.

Supervised Cluster Evaluation

Other measures and variants

Some further evaluation measures:

- Adjustment for chance is general principle,

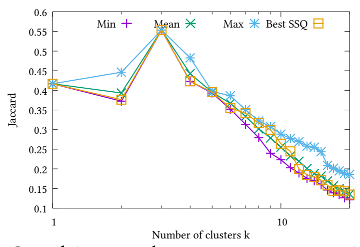
$$\text{Adjusted Index} = \frac{\text{Index}-E[\text{Index}]}{\text{Optimal Index}-E[\text{Index}]}$$

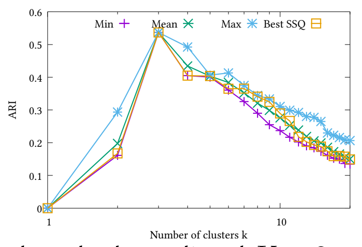
  For example Adjusted Rand Index [HA85] or Adjusted Mutual Information [VEB10]
- B-Cubed evaluation [BB98]
- Set matching purity [ZK01] and F1 [SKK00]
- Edit distance [PL02]
- Visual comparison of multiple clusterings [Ach+12]
- Gini-based evaluation [Sch+15]

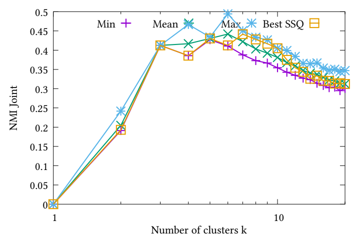
Supervised Cluster Evaluation

Examples: Mouse data

Revisiting k-means on the mouse data:







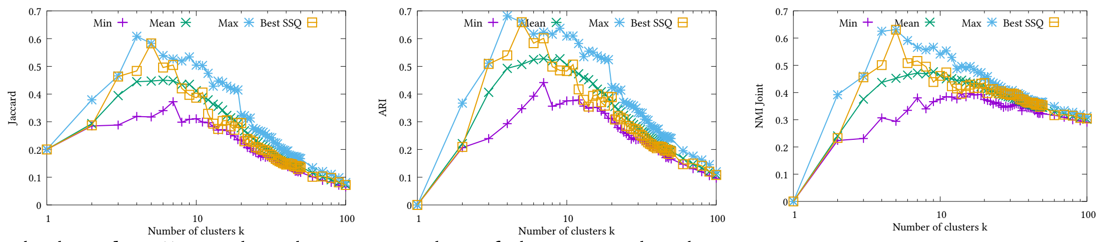
On this toy data set, unsupervised methods predicted $K = 3$.

go back

Supervised Cluster Evaluation

Examples: Tutorial's Wikipedia data set

Revisiting k-means on the Wikipedia data set (c.f., tutorials):



The best $k = 5$ matches the true number of clusters in this data set.
Unsupervised measures would have preferred $k = 2$.

go back

Text Clustering

Conclusions

Clustering text data is hard because:

- Preprocessing (TF-IDF etc.) has major impact
  (But we do not know which preprocessing is "correct" or "best")
- Text is high-dimensional, and our intuition of "distance" and "density" do not work well
- Text is sparse, and many clustering assume dense, Gaussian data.
- Text is noisy, and many documents may not be part of a cluster at all.
- Some cases can be handled with biclustering or frequent itemset mining.
- The (proper) evaluation of clustering is very difficult: [JD88]

  The validation of clustering structures is the most difficult and frustrating part of cluster analysis.
  Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage.

➨ We need methods *designed* for text: topic modeling

References I

[Ach+12] E. Achtert, S. Goldhofer, H. Kriegel, E. Schubert, and A. Zimek. "Evaluation of Clusterings - Metrics and Visual Support". In: *IEEE 28th International Conference on Data Engineering (ICDE 2012)*. 2012, pp. 1285–1288.

[Agg+99] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. "Fast Algorithms for Projected Clustering". In: *Proc. ACM SIGMOD International Conference on Management of Data*. 1999, pp. 61–72.

[Agr+98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications". In: *Proc. ACM SIGMOD International Conference on Management of Data*. 1998, pp. 94–105.

[Aka77] H. Akaike. "On entropy maximization principle". In: *Applications of Statistics*. 1977, pp. 27–41.

[And73] M. R. Anderberg. "Cluster analysis for applications". In: Probability and mathematical statistics. Academic Press, 1973. Chap. Hierarchical Clustering Methods, pp. 131–155. ISBN: 0120576503.

[AS94] R. Agrawal and R. Srikant. "Fast Algorithms for Mining Association Rules in Large Databases". In: *Proc. VLDB*. 1994, pp. 487–499.

[AV06] D. Arthur and S. Vassilvitskii. "How slow is the k-means method?" In: *Symposium on Computational Geometry*. 2006, pp. 144–153.

[AV07] D. Arthur and S. Vassilvitskii. "k-means++: the advantages of careful seeding". In: *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2007, pp. 1027–1035.

[Ban+05] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. "Clustering with Bregman Divergences". In: *J. Machine Learning Research* 6 (2005), pp. 1705–1749.

References II

[BB98] A. Bagga and B. Baldwin. "Entity-Based Cross-Document Coreferencing Using the Vector Space Model". In: *COLING-ACL*. 1998, pp. 79–85.

[Bey+99] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. "When Is "Nearest Neighbor" Meaningful?" In: *International Conference on Database Theory (ICDT)*. 1999, pp. 217–235.

[BH75] F. B. Baker and L. J. Hubert. "Measuring the Power of Hierarchical Cluster Analysis". In: *Journal American Statistical Association* 70.349 (1975), pp. 31–38.

[Boc07] H. Bock. "Clustering Methods: A History of k-Means Algorithms". In: *Selected Contributions in Data Analysis and Classification*. Ed. by P. Brito, G. Cucumel, P. Bertrand, and F. Carvalho. Springer, 2007, pp. 161–172.

[Cad+03] I. V. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. "Model-Based Clustering and Visualization of Navigation Patterns on a Web Site". In: *Data Min. Knowl. Discov.* 7.4 (2003), pp. 399–424.

[CC00] Y. Cheng and G. M. Church. "Biclustering of Expression Data". In: *ISMB*. 2000, pp. 93–103.

[CH74] T. Caliński and J. Harabasz. "A dendrite method for cluster analysis". In: *Communications in Statistics* 3.1 (1974), pp. 1–27.

[DB79] D. Davies and D. Bouldin. "A cluster separation measure". In: *Pattern Analysis and Machine Intelligence* 1 (1979), pp. 224–227.

[DD09] M. M. Deza and E. Deza. *Encyclopedia of Distances*. 3rd. Springer, 2009. ISBN: 9783662443415.

References III

[Def77] D. Defays. "An Efficient Algorithm for the Complete Link Cluster Method". In: *The Computer Journal* 20.4 (1977), pp. 364–366.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood from Incomplete Data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 39.1 (1977), pp. 1–31.

[DM01] I. S. Dhillon and D. S. Modha. "Concept Decompositions for Large Sparse Text Data Using Clustering". In: *Machine Learning* 42.1/2 (2001), pp. 143–175.

[Dun73] J. C. Dunn. "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters". In: *Journal of Cybernetics* 3.3 (1973), pp. 32–57.

[Dun74] J. C. Dunn. "Well separated clusters and optimal fuzzy partitions". In: *Journal of Cybernetics* 4 (1974), pp. 95–104.

[FM83] E. B. Fowlkes and C. L. Mallows. "A Method for Comparing Two Hierarchical Clusterings". In: *Journal American Statistical Association* 78.383 (1983), pp. 553–569.

[For65] E. W. Forgy. "Cluster analysis of multivariate data: efficiency versus interpretability of classifications". In: *Biometrics* 21 (1965), pp. 768–769.

[HA85] L. Hubert and P. Arabie. "Comparing partitions". In: *Journal of Classification* 2.1 (1985), pp. 193–218.

[Har75] J. A. Hartigan. *Clustering Algorithms*. New York, London, Sydney, Toronto: John Wiley&Sons, 1975.

[HL76] L. J. Hubert and J. R. Levin. "A general statistical framework for assessing categorical clustering in free recall." In: *Psychological Bulletin* 83.6 (1976), pp. 1072–1080.

References IV

[Hof99] T. Hofmann. "Probabilistic Latent Semantic Indexing". In: *ACM SIGIR*. 1999, pp. 50–57.

[Hou+10] M. E. Houle, H. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Can Shared-Neighbor Distances Defeat the Curse of Dimensionality?" In: *Int. Conf. on Scientific and Statistical Database Management (SSDBM)*. 2010, pp. 482–500.

[HPY00] J. Han, J. Pei, and Y. Yin. "Mining Frequent Patterns without Candidate Generation". In: *Proc. SIGMOD*. 2000, pp. 1–12.

[HW79] J. A. Hartigan and M. A. Wong. "Algorithm AS 136: A k-means clustering algorithm". In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* (1979), pp. 100–108.

[JD88] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Englewood Cliffs: Prentice Hall, 1988.

[JZM04] X. Jin, Y. Zhou, and B. Mobasher. "Web usage mining based on probabilistic latent semantic analysis". In: *ACM SIGKDD*. 2004, pp. 197–205.

[KKK04] P. Kröger, H. Kriegel, and K. Kailing. "Density-Connected Subspace Clustering for High-Dimensional Data". In: *Proc. of the Fourth SIAM International Conference on Data Mining*. 2004, pp. 246–256.

[KR90] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analyis*. John Wiley&Sons, 1990. ISBN: 9780471878766.

[KSZ16] H. Kriegel, E. Schubert, and A. Zimek. "The (black) art of runtime evaluation: Are we comparing algorithms or implementations?" In: *Knowledge and Information Systems (KAIS)* (2016), pp. 1–38.

[Llo82] S. P. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–136.

References V

[LW67] G. N. Lance and W. T. Williams. "A General Theory of Classificatory Sorting Strategies. 1. Hierarchical Systems". In: *The Computer Journal* 9.4 (1967), pp. 373–380.

[Mac67] J. MacQueen. "Some Methods for Classification and Analysis of Multivariate Observations". In: *5th Berkeley Symposium on Mathematics, Statistics, and Probabilistics.* Vol. 1. 1967, pp. 281–297.

[Mei03] M. Meila. "Comparing Clusterings by the Variation of Information". In: *Computational Learning Theory (COLT).* 2003, pp. 173–187.

[Mei05] M. Meila. "Comparing Clusterings – An Axiomatic View". In: *Int. Conf. Machine Learning (ICML).* 2005, pp. 577–584.

[Mei12] M. Meilă. "Local equivalences of distances between clusterings–a geometric perspective". In: *Machine Learning* 86.3 (2012), pp. 369–389.

[Mou+14] D. Moulavi, P. A. Jaskowiak, R. J. G. B. Campello, A. Zimek, and J. Sander. "Density-based Clustering Validation". In: *SIAM SDM.* 2014, pp. 839–847.

[MRS08] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval.* Cambridge University Press, 2008. ISBN: 978-0-521-86571-5. URL: `http://nlp.stanford.edu/IR-book/`.

[MS01] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing.* MIT Press, 2001. ISBN: 978-0-262-13360-9.

[PBM04] M. K. Pakhira, S. Bandyopadhyay, and U. Maulik. "Validity index for crisp and fuzzy clusters". In: *Pattern Recognition* 37.3 (2004), pp. 487–501.

References VI

[Phi02] S. J. Phillips. "Acceleration of K-Means and Related Clustering Algorithms". In: *ALENEX'02.* 2002, pp. 166–177.

[PL02] P. Pantel and D. Lin. "Document clustering with committees". In: *ACM SIGIR.* 2002, pp. 199–206.

[PM00] D. Pelleg and A. Moore. "X-means: Extending k-means with efficient estimation of the number of clusters". In: *Proceedings of the 17th International Conference on Machine Learning (ICML).* Vol. 1. 2000, pp. 727–734.

[Ran71] W. M. Rand. "Objective criteria for the evaluation of clustering methods". In: *Journal American Statistical Association* 66.336 (1971), pp. 846–850.

[Rou87] P. J. Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[Sch+15] E. Schubert, A. Koos, T. Emrich, A. Züfle, K. A. Schmid, and A. Zimek. "A Framework for Clustering Uncertain Data". In: *Proceedings of the VLDB Endowment* 8.12 (2015), pp. 1976–1979.

[Sch78] G. Schwarz. "Estimating the dimension of a model". In: *The Annals of Statistics* 6.2 (1978), pp. 461–464.

[Sib73] R. Sibson. "SLINK: An Optimally Efficient Algorithm for the Single-Link Cluster Method". In: *The Computer Journal* 16.1 (1973), pp. 30–34.

[SKK00] M. Steinbach, G. Karypis, and V. Kumar. "A comparison of document clustering techniques". In: *KDD workshop on text mining.* Vol. 400. 2000, pp. 525–526.

[Sne57] P. H. A. Sneath. "The Application of Computers to Taxonomy". In: *Journal of General Microbiology* 17 (1957), pp. 201–226.

References VII

[Ste56] H. Steinhaus. "Sur la division des corp materiels en parties". In: *Bull. Acad. Polon. Sci* 1 (1956), pp. 801–804.

[VEB10] N. X. Vinh, J. Epps, and J. Bailey. "Information Theoretic Measures for Clustering Comparison: Variants, Properties, Normalization and Correction for Chance". In: *J. Machine Learning Research* 11 (2010), pp. 2837–2854.

[YH02] A. Ypma and T. Heskes. "Automatic Categorization of Web Pages and User Clustering with Mixtures of Hidden Markov Models". In: *Workshop WEBKDD.* 2002, pp. 35–49.

[ZK01] Y. Zhao and G. Karypis. *Criterion Functions for Document Clustering: Experiments and Analysis.* Tech. rep. 01-40. University of Minnesota, Department of Computer Science, 2001.

[ZSK12] A. Zimek, E. Schubert, and H. Kriegel. "A Survey on Unsupervised Outlier Detection in High-Dimensional Numerical Data". In: *Statistical Analysis and Data Mining* 5.5 (2012), pp. 363–387.

[ZXF08] Q. Zhao, M. Xu, and P. Fränti. "Knee Point Detection on Bayesian Information Criterion". In: *IEEE International Conference on Tools with Artificial Intelligence (ICTAI).* 2008, pp. 431–438.

Topic Modeling

Motivation

Find the latent structure in a text corpus that:

- resembles “topics” (also “concepts”)
- best summarize the collection
- is based on statistical patterns
- are obscured by synonyms, homonyms, stopwords, …
- may overlap

Similar to clustering, but with a slightly different “mindset”:

- In clustering, the emphasis is on the data points / documents
- In topic modeling, the emphasis is on the topics / clusters themselves

Literature

General introduction to LDA:
D. M. Blei. “Probabilistic topic models”. In: *Commun. ACM* 55.4 (2012), pp. 77–84

Lecture by David Blei:
`http://videolectures.net/mlss09uk_blei_tm/`

Probabilistic graphical modeling textbook:
D. Koller and N. Friedman. *Probabilistic Graphical Models - Principles and Techniques.* MIT Press, 2009. ISBN: 978-0-262-01319-2

Topic modeling chapter (17) of this textbook:
C. Zhai and S. Massung. *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining.* New York, NY, USA: Association for Computing Machinery and Morgan & Claypool, 2016. ISBN: 978-1-97000-117-4

LSI/LSA: Topics via Matrix Factorization

Latent Semantic Indexing (LSI) [Fur+88; Dee+90] was developed to improve *information retrieval*.
Also called Latent Semantic Analysis (LSA).

In information retrieval, synonymy and polysemy are a challenge:

- exact search will not find synonyms
- exact search will include polynyms and homonyms

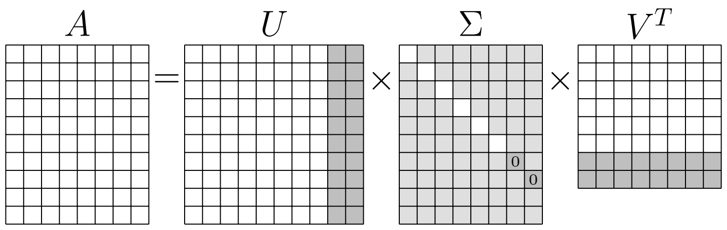
Idea: identify “factors” that can contain multiple words, or parts of a word
Factors are a *lower-dimensional* representation of the document.

Factor analysis of the document-term matrix:

- similarity of words based on the documents they cooccur in
- similarity of documents based on the words they contain

Topics via Matrix Factorization

Recall Singular Value Decomposition (SVD):



- A: document-term matrix
- U: document-topic map ("topic distribution")
- Σ: topic importance
- V: term-topic map (V^T: "term distribution")

By truncating the matrix to k topics, we get the best (least-squares) approximation.

Topics via Matrix Factorization II

Complexity of SVD on a $m \times n$ matrix is: $\mathcal{O}(\min\{mn^2, m^2n\}) = \mathcal{O}(mn \cdot \min\{m,n\})$

We can approximate this in $\mathcal{O}(k^2 \cdot \min\{m,n\})$ using Monte-Carlo sampling [FKV04] if we only need k components, to be more efficient.

➨ If we do a stochastic approximation, can we use a probabilistic model directly?

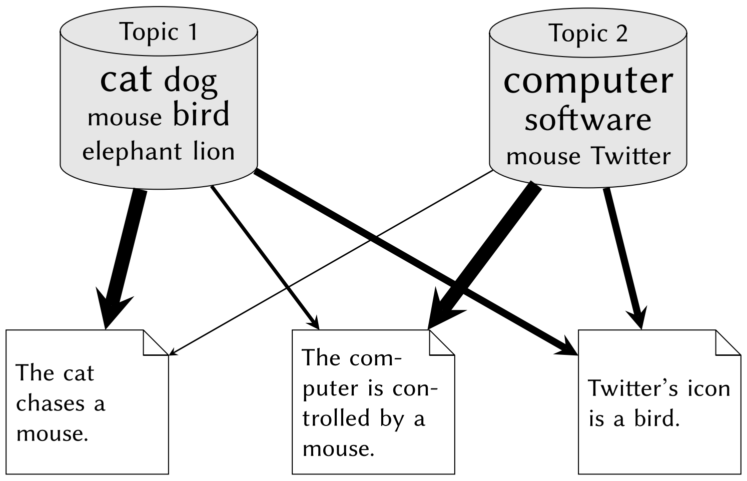
U $\sim$ a topic distribution for every document
$V^T \sim$ a word distribution for every topic

➨ model these as probabilities $P(T_k \mid d_i)$ and $P(w_j \mid T_i)$

Note: SVD does not yield probabilities, but factors can contain negative values.

Topic Modeling

From Matrix Factors to Topic Models



Topic Modeling

Probabilistic Mathematical Model

Basic idea of probabilistic topic modeling:

Every document d_i is a mixture of topics T_k (with $\theta_{i,k} \geq 0$):

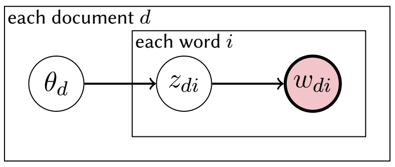
$$\sum\nolimits_k P(d_i \in T_k) = \sum\nolimits_k \underbrace{\theta_{i,k}}_{\text{"topic distribution" of document } i} = 1$$

Every word $w_{i,j}$ is drawn from one of the documents' topics:

$$P(w_{i,j}) = \sum\nolimits_k P(w_{i,j} \mid d_i \in T_k) P(d_i \in T_k) = \sum\nolimits_k \underbrace{\varphi_k(w_{i,j})}_{\text{"word distribution" of topic } k} \theta_{i,k}$$

Probabilistic Topic Modeling

pLSI: probabilistic Latent Semantic Indexing [Hof99b; Hof99a]



Where z_{di} is the topic of the ith word in document d.

This assumes conditional independence given an unobserved topic t: [BNJ03]

$$P(w, d) = P(d) \sum_t P(w \mid t) P(t \mid d)$$

- See the EM chapter for an EM-algorithm for the multinomial distribution.

○ Prior ○ Hidden var. ● Observed var. → Dependency □ Repeated element

Probabilistic Topic Modeling

Probabilistic generative model

The model assumes our data set was generated by a process like this:

1. For every topic t, sample a word distribution φ_t
2. For every document d, sample a topic distribution θ_d
3. For every document d, generate l words $i = 1 \ldots l$:
   - 3.1 Sample a topic z_{di} from θ_d
   - 3.2 Sample a word w_{di} from the distribution $\varphi_{z_{di}}$

Note: this is *not* what we *do*, but our underlying assumptions.

Probabilistic Topic Modeling

Likelihood of PLSI

In PLSI, we model a document d as mixtures of k topics:

$$\underbrace{P(w \mid d)}_{\substack{\text{Word } w \text{ in} \\ \text{document (model) } d}} = \sum\nolimits_t \underbrace{\theta_{d,t}}_{\substack{\text{Weight of topic } t \\ \text{in document } d}} \underbrace{\varphi_{t,w}}_{\substack{\text{Prob. word } w \\ \text{in topic } t}}$$

$$\underbrace{\log P(d)}_{\substack{\text{Loglikelihood} \\ \text{of document } d}} = \sum\nolimits_w \log \; P(w \mid d)$$

$$\underbrace{\log P(D)}_{\substack{\text{Loglikelihood} \\ \text{of all documents}}} = \sum\nolimits_d \log P(d)$$

$$\log P(D) = \sum\nolimits_d \sum\nolimits_w \log \left[\sum\nolimits_t \theta_{d,t} \; \varphi_{t,w} \right]$$

Log-loss function to maximize

Probabilistic Topic Modeling

EM-Algorithm for PLSI

Normalize to a proper probability distribution

Expectation-Step (estimate $z_{d,w}$ from previous θ, φ):

$$\underbrace{P(z_{d,w} = t)}_{\substack{\text{Prob. that word } w \\ \text{is in topic } t}} \propto \underbrace{\theta_{d,t}}_{\substack{\text{Weight of topic } t \\ \text{in document } d}} \underbrace{\varphi_{t,w}}_{\substack{\text{Prob. word } w \\ \text{in topic } t}} \qquad \text{s.t. } \forall_{d,w} : \sum\nolimits_t P(z_{d,w} = t) = 1$$

Maximization-Step (optimize θ, $\varphi_{t,w}$ from $z_{d,w}$):

$$\theta_{d,t} \propto \sum\nolimits_w \mathrm{df}_{w,d} \, P(z_{d,w} = t) \qquad \text{s.t. } \forall_d : \sum\nolimits_t \theta_{d,t} = 1$$

$$\varphi_{t,w} \propto \sum\nolimits_d \mathrm{df}_{w,d} \, P(z_{d,w} = t) \qquad \text{s.t. } \forall_t : \sum\nolimits_w \varphi_{t,w} = 1$$

Note: because of θ_d, which depends on the document d,
the topic estimation of a word can be different in different documents!

This is the Maximum-Likelihood Estimation (MLE).

Probabilistic Topic Modeling

Incorporating prior knowledge

PLSI will model all words, including stopwords!
This causes some problems, we therefore can:

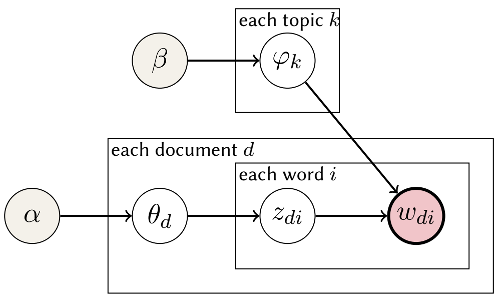
- Remove stopwords
- Add a “background” topic (c.f. [ZM16])
- Use Maximum-a-posterior (MAP) estimation, with a prior word distribution

$$\varphi_{t,w} \propto \sum\nolimits_d \mathrm{df}_{w,d} \, P(z_{d,w} = t) + \mu\varphi'_w \qquad \text{s.t. } \forall_t : \sum\nolimits_w \varphi_{t,w} = 1$$

Where $\mu \in [0; \infty]$ controls the strength of prior information,
and φ' is the prior word distribution. [ZM16]

Probabilistic Topic Modeling

LDA: Latent Dirichlet Allocation [BNJ01; BNJ03; Ble12]



For every word, we first draw a topic z_{di}, then draw a word w_{di} from this topic.
Topic and word distributions use *"sparse"* Dirichlet distributions.

◯ Prior ◯ Hidden var. ● Observed var. → Dependency □ Repeated element

Latent Dirichlet Allocation

Generative Process of LDA

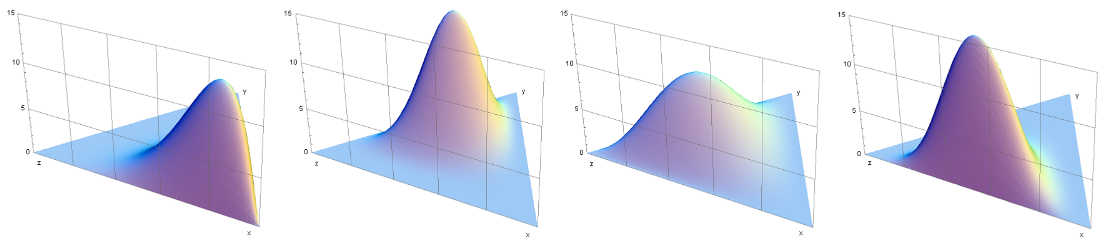
LDA assumes, documents are generated as follows:

1. For each document d, draw a topic distribution θ_d from a Dirichlet distribution
$$\theta_d \sim \text{Dirichlet}(\alpha).$$
2. For each topic k, draw a word distribution φ_k from a Dirichlet distribution
$$\varphi_k \sim \text{Dirichlet}(\beta).$$
3. For each word w_{di} in each document d draw a topic z_{di} from the multinomial θ_d
$$z_{di} \sim \text{Discrete}(\theta_d).$$
4. Draw a word from the multinomial $\varphi_{z_{di}}$
$$w_{di} \sim \text{Discrete}(\varphi_{z_{di}}).$$

Latent Dirichlet Allocation

Dirichlet Distribution

The Dirichlet prior of LDA – density with $k = 3$, $\alpha > 1$:



But: we will be using $\alpha < 1$, where this distribution becomes sparse.

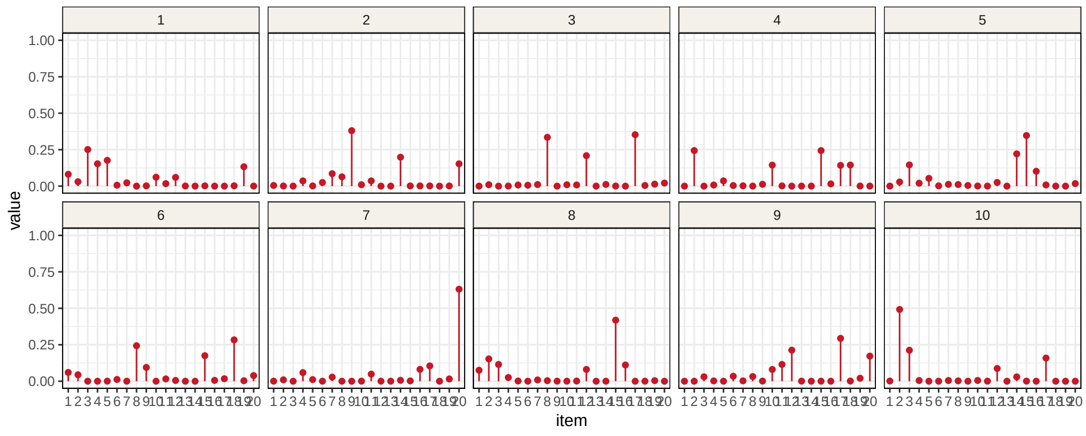
PD Image from Wikipedia, `https://commons.wikimedia.org/wiki/File:Dirichlet_distributions.png`

Latent Dirichlet Allocation

Dirichlet Distribution

Most values are ≈ 0

The Dirichlet prior of LDA – samples with $k = 20$, $\alpha = 0.2$



Latent Dirichlet Allocation

Likelihood of LDA

LDA adds the Dirichlet prior to model the likeliness of word and topic distributions.

$$\underbrace{\log P(d)}_{\text{Loglikelihood of document } d} = \int \sum_w \log \sum_t \underbrace{\theta_{d,t}}_{\text{Weight of topic } t \text{ in document } d} \underbrace{\varphi_{t,w}}_{\text{Prob. word } w \text{ in topic } t} \underbrace{P(\varphi_t \mid \alpha)\, \mathrm{d}\varphi_t}_{\text{Likelihood of word distribution}}$$

$$\log P(D) = \int \sum_d \sum_w \log \left[\sum_t \theta_{d,t} \varphi_{t,w} \right] \prod_t P(\varphi_t \mid \alpha)\, \mathrm{d}\varphi_1 \cdots \mathrm{d}\varphi_k$$

A model is *better* if the word distributions match our Dirichlet prior better!

Latent Dirichlet Allocation

Computation of LDA

We do not generate random documents, but we need to compute the likelihood of a document, and optimize (hyper-) parameters to best explain the documents.

We cannot solve this exactly, but we need to approximate this.

- Variational inference [BNJ01; BNJ03]
- Gibbs sampling [PSD00; Gri02]
- Expectation propagation [ML02]
- Collapsed Gibbs sampling [GS04]
- Collapsed variational inference [TNW06]
- Sparse collapsed Gibbs sampling [YMM09]
- Metropolis-Hastings-Walker sampling [Li+14]

Gibbs Sampling

Monte-Carlo Methods

We need to estimate complex functions that we cannot handle analytically.
Estimates of a function $f(x)$ usually look like this:

$$E[f(x)] = \sum\nolimits_z f(y)p(y) \simeq \int_y f(y)p(y)\,\mathrm{d}y$$

where $p(y)$ is the likelihood of the input parameters x being $x = y$.

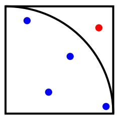
Monte-Carlo methods estimate from a sample set $Y = \{y^{(i)}\}$:

$$E[f(x)] \approx \frac{1}{|Y|} \sum\nolimits_{y^{(i)}} f(y^{(i)})$$

No $p(x)$, because the $y^{(i)}$ are observed with probability $p(y^{(i)})$

Important: we *require* the $y^{(i)}$ to occur with $p(y^{(i)})$.

Example: Estimate $\frac{\pi}{4}$ by choosing points in the unit square uniformly, and testing if they are within the unit circle (here, uniform is okay).



Gibbs Sampling

Markov Chains

Monte Carlo is simple, but how do we get such $y^{(i)}$ according to their probabilities $p(y^{(i)})$?
In a Markov process, the new state $y^{(t+1)}$ only depends on the previous state $y^{(t)}$:

$$P(y^{(t+1)} \mid y^{(1)}, \ldots, y^{(t)}) = P(y^{(t+1)} \mid y^{(t)})$$

We need to design a *transition function* g such that $y^{(t+1)} = g(y^{(t)})$ and $p(y^{(t+1)})$ as desired.

$$\underbrace{y^{(0)} \xrightarrow[g]{} y^{(1)} \xrightarrow[g]{} y^{(2)} \xrightarrow[g]{}}_{\text{The first } B \text{ are often ignored}} \cdots \xrightarrow[g]{} \underbrace{y^{(t)} \xrightarrow[g]{} y^{(t+1)} \xrightarrow[g]{} \cdots}_{\text{These occur with } p(y^{(i)})}$$

For g, we can use, e.g., Gibbs sampling. We then can estimate our hidden variables!
Because of autocorrelation, it is common to use only every Lth sample.
(We require P above to be ergodic, but omit details in this lecture.)

A really nice introduction to Markov-Chain-Monte-Carlo (MCMC) and Gibbs sampling can be found in [RH10].
A more formal introduction is in the textbook [Bis07].

Gibbs Sampling

Updating variables incrementally

Assume that our state $y^{(t)}$ is a vector with $k > 1$ components,
we can update one variable at a time for $i = 1 \ldots k$:

y_i omitted

$$y_i^{(t+1)} \sim P(Y_i \mid \underbrace{y_1^{(t+1)}, \ldots, y_{i-1}^{(t+1)}}_{\text{already updated}}, \underbrace{y_{i+1}^{(t)}, \ldots, y_k^{(t)}}_{\text{not yet updated}})$$

Our function g then is to do this for each $i = 1 \ldots k$.

Informally: in every iteration ($t \to t+1$), for every variable i, we choose a new value $y_i^{(t+1)}$ *randomly*, but we prefer values more likely given the current state of the other variables.

More likely values of y will be more likely returned (even with the desired likelihood of $p(y)$).

Gibbs Sampling

Benefits and details

$P(Y_i \mid y_1, \ldots)$ may not depend on all y_j, but only on the “Markov blanket”.
(Markov blanket: parents, children, and other parents of the node’s children in the diagram.)

Sometimes we can also “integrate out” some y_j to further simplify P.
This is also called a “collapsed” Gibbs sampler.

If we have a conjugate prior (e.g., Beta for Bernoulli, Dirichlet for Multinomial),
then we get the same family (but different parameters) a posterior,
which usually yields much simpler equations.

Gibbs Sampling

Collapsed Gibbs sampler [GS04]

Computing this is very expensive

We need to draw the topic of each word according to:

$$P(z_i = t \mid w, d, \ldots) \propto \prod_k \underbrace{P(\varphi_k \mid \beta)}_{P(\text{topic})} \prod_d \underbrace{P(\theta_d \mid \alpha)}_{P(\text{document})} \prod_w \underbrace{P(z_{dw} \mid \theta_d) P(w_{dw} \mid \varphi_{z_{dw}})}_{P(\text{word given topic})}$$

After integrating out φ and θ, we get the word-topic probability:

$$P(z_i = t \mid w, d, \ldots) \propto \prod\nolimits_t \left[\Gamma(n_{td} + \alpha_k) \cdot \frac{\Gamma(n_{tw}+\beta_w)}{\Gamma(\sum_{w'} n_{tw'}+\beta_{w'})}\right]$$

By exploiting properties of the Γ function, we can simplify this to:

$$P(z_i = t \mid w, d, \ldots) \propto (n_{td}^{-di} + \alpha_t) \cdot \frac{(n_{tw}^{-di}+\beta_w)}{n_t^{-di}+\sum_{w'} \beta_{w'}}$$

where n_{td}^{-di}, n_{tw}^{-di}, and n_t^{-di} are the number of occurrences of a topic-document assignment, topic-word assignment, or topic, *ignoring* the current word w_{di} and its topic assignment z_{di}.

A detailed derivation can be found in Appendix D of [Cha11].

Latent Dirichlet Allocation

Inference with Gibbs Sampling

Putting everything together:

1. Initialization:
   1.1 Choose prior parameters α and β.
   1.2 For every document and word, choose z_{di} randomly.
   1.3 Initialize n_{td}, n_{tw}, and n_t.
2. For every Markov-Chain iteration $j = 1 \ldots I$:
   2.1 For every document d and word w_{di}:
      2.1.1 Remove old z_{di}, w_{di} from n_{td}, n_{tw}, and n_t
      2.1.2 Sample a new random topic z_{di}
      2.1.3 Update n_{td}, n_{tw}, and n_t with new z_{di}, re-add w_{di}.
   2.2 If $j \geq B$ (burn in) and only every Lth sample (decorrelation):
      2.2.1 Monte-Carlo update all θ_d, φ_k from z_{di}

We try putting words in different topics randomly

The more often we see a topic, the more relevant it is.

Probabilistic Topic Modeling

More complicated models

Many variations have been proposed.

- We can vary the prior assumptions (to draw θ, φ).
  E.g. Rethinking LDA: Why priors matter [WMM09]
  But conjugate priors like Dirichlet-Multinomial are easier to compute.
- Also learn the number of topics, α, and β (may require labeled data).
- Hierarchical Dirichlet Processes [Teh+06]
- Pitman-Yor and Poisson Dirichlet Processes [PY97; SN10]
- Correlated Topic Models [BL05]
- Application to other domains (instead of text).

Evaluation of Topic Models

"Reading Tea Leaves" [Cha+09; LNB14]

Topic model evaluation is difficult:

> *"There is a disconnect between how topic models are evaluated and why we expect topic models to be useful." – David Blei [Ble12]*

- Often evaluated with a secondary task (e.g., classification, IR) [Wal+09]
- By the ability to explain held out documents with existing clusters [Wal+09]
  (A document is "well explained" if it has a high probability in the model)
- Manual inspection of the most important words in each topic
- Word intrusion task [Cha+09]
  (Can a user identify a word that was artificially injected into the most important words?)
- Topic intrusion task [Cha+09]
  (Can the user identify a topic that doesn't apply to a test document?)

References I

[Bis07] C. M. Bishop. *Pattern recognition and machine learning, 5th Edition.* Information science and statistics. Springer, 2007. ISBN: 9780387310732.

[BL05] D. M. Blei and J. D. Lafferty. "Correlated Topic Models". In: *Neural Information Processing Systems, NIPS.* 2005, pp. 147–154.

[Ble12] D. M. Blei. "Probabilistic topic models". In: *Commun. ACM* 55.4 (2012), pp. 77–84.

[BNJ01] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent Dirichlet Allocation". In: *Neural Information Processing Systems, NIPS.* 2001, pp. 601–608.

[BNJ03] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent Dirichlet Allocation". In: *J. Machine Learning Research* 3 (2003), pp. 993–1022.

[Cha+09] J. Chang, J. L. Boyd-Graber, S. Gerrish, C. Wang, and D. M. Blei. "Reading Tea Leaves: How Humans Interpret Topic Models". In: *Neural Information Processing Systems, NIPS.* 2009, pp. 288–296.

[Cha11] J. Chang. "Uncovering, Understanding, and Predicting Links". PhD thesis. Princeton University, 2011.

[Dee+90] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. "Indexing by Latent Semantic Analysis". In: *JASIS* 41.6 (1990), pp. 391–407.

[FKV04] A. M. Frieze, R. Kannan, and S. Vempala. "Fast monte-carlo algorithms for finding low-rank approximations". In: *J. ACM* 51.6 (2004), pp. 1025–1041.

References II

[Fur+88] G. W. Furnas, S. C. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. "Information Retrieval using a Singular Value Decomposition Model of Latent Semantic Structure". In: *ACM SIGIR*. 1988, pp. 465–480.

[Gri02] T. L. Griffiths. *Gibbs sampling in the generative model of latent dirichlet allocation*. Tech. rep. Stanford University, 2002.

[GS04] T. L. Griffiths and M. Steyvers. "Finding scientific topics". In: *Proceedings of the National Academy of Sciences* 101.suppl 1 (2004), pp. 5228–5235.

[Hof99a] T. Hofmann. "Learning the Similarity of Documents: An Information-Geometric Approach to Document Retrieval and Categorization". In: *Neural Information Processing Systems, NIPS*. 1999, pp. 914–920.

[Hof99b] T. Hofmann. "Probabilistic Latent Semantic Indexing". In: *ACM SIGIR*. 1999, pp. 50–57.

[KF09] D. Koller and N. Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009. ISBN: 978-0-262-01319-2.

[Li+14] A. Q. Li, A. Ahmed, S. Ravi, and A. J. Smola. "Reducing the sampling complexity of topic models". In: *ACM SIGKDD*. 2014, pp. 891–900.

[LNB14] J. H. Lau, D. Newman, and T. Baldwin. "Machine Reading Tea Leaves: Automatically Evaluating Topic Coherence and Topic Model Quality". In: *European Chapter of the Association for Computational Linguistics, EACL*. 2014, pp. 530–539.

[ML02] T. P. Minka and J. D. Lafferty. "Expectation-Propogation for the Generative Aspect Model". In: *UAI '02*. 2002, pp. 352–359.

References III

[PSD00] J. K. Pritchard, M. Stephens, and P. Donnelly. "Inference of Population Structure Using Multilocus Genotype Data". In: *Genetics* 155.2 (2000), pp. 945–959. ISSN: 0016-6731.

[PY97] J. Pitman and M. Yor. "The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator". In: *Ann. Probab.* 25.2 (Apr. 1997), pp. 855–900.

[RH10] P. Resnik and E. Hardisty. *Gibbs sampling for the uninitiated*. Tech. rep. CS-TR-4956. University of Maryland, 2010.

[SN10] I. Sato and H. Nakagawa. "Topic models with power-law using Pitman-Yor process". In: *ACM SIGKDD*. 2010, pp. 673–682.

[Teh+06] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. "Hierarchical Dirichlet Processes". In: *J. American Statistical Association* 101.476 (2006), pp. 1566–1581.

[TNW06] Y. W. Teh, D. Newman, and M. Welling. "A Collapsed Variational Bayesian Inference Algorithm for Latent Dirichlet Allocation". In: *Neural Information Processing Systems, NIPS*. 2006, pp. 1353–1360.

[Wal+09] H. M. Wallach, I. Murray, R. Salakhutdinov, and D. M. Mimno. "Evaluation methods for topic models". In: *International Conference on Machine Learning, ICML*. 2009, pp. 1105–1112.

[WMM09] H. M. Wallach, D. M. Mimno, and A. McCallum. "Rethinking LDA: Why Priors Matter". In: *Neural Information Processing Systems, NIPS*. 2009, pp. 1973–1981.

[YMM09] L. Yao, D. M. Mimno, and A. McCallum. "Efficient methods for topic model inference on streaming document collections". In: *ACM SIGKDD*. 2009, pp. 937–946.

References IV

[ZM16] C. Zhai and S. Massung. *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. New York, NY, USA: Association for Computing Machinery and Morgan & Claypool, 2016. ISBN: 978-1-97000-117-4.

Word Embeddings

Motivation

Consider these two sentences:[8]

$$\begin{tikzcd}
 & \texttt{Obama} \arrow[d, leftrightarrow] & \texttt{speaks} \arrow[d, leftrightarrow] & \texttt{to} & \texttt{the} & \texttt{media} \arrow[dl, leftrightarrow] & \texttt{in} & \texttt{Illinois} \arrow[dl, leftrightarrow] & \texttt{.} \\
\texttt{The} & \texttt{President} & \texttt{greets} & \texttt{the} & \texttt{press} & \texttt{in} & \texttt{Chicago} & \texttt{.} &
\end{tikzcd}$$

Cosine similarity: 0, if stop words were removed.

Want to recognize:

Obama	$\sim$	President
speaks	$\sim$	greets
press	$\sim$	media
Illinois	$\sim$	Chicago

[8]Example taken from [Kus+15]

Word Embeddings

Motivation

The bag-of-words representation does not capture word similarity:

For example the words Obama and President:

$$\begin{array}{rl}
\texttt{Obama} & = (\ 0\ ,\ 0\ ,\ 0\ ,\ 1\ ,\ 0\ ,\ \dots\ ,\ 0) \\
\texttt{President} & = (\ 0\ ,\ 1\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ \dots\ ,\ 0) \\
\texttt{Obama} \cdot \texttt{President} & = (\ 0\ , 0\cdot 1,\ 0\ , 1\cdot 0,\ 0\ ,\ \dots\ ,\ 0) = 0
\end{array}$$

Because of this, the documents are completely dissimilar (except for stopwords):

$$\mathrm{sim}(\{\texttt{Obama}, \texttt{speaks}, \texttt{press}, \texttt{Illinois}\}, \{\texttt{President}, \texttt{greets}, \texttt{media}, \texttt{Chicago}\}) = 0$$

➨ We want a word representation where $0 \ll \mathrm{sim}(\texttt{Obama}, \texttt{President}) < 1$.

Preferrably also of lower dimensionality (100–500) than our vocabulary!

Contextual information

What is a Wampimuk?[10]

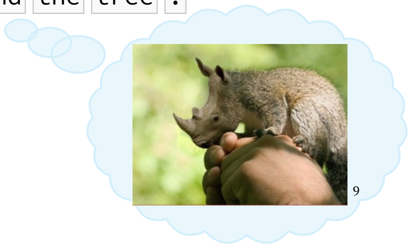
Humans infer meaning from the context [MR01]:

`He` `filled` `the` `wampimuk` `,` `passed` `it` `around` `and` `we` `all` `drunk` `some` `.`

➨ Probably a drink?

`We` `found` `a` `cute` `,` `hairy` `wampimuk` `sleeping` `behind` `the` `tree` `.`

➨ Probably an animal?



We want computers to be able to make such inferences!

[9]Image from [LBB14], but probably originally internet folklore.

[10]This example is probably from Marco Baroni ca. 2011 to illustrate the distributional hypothesis of [MR01], but is frequently (incorrectly?) attributed to [MR01].

Vector representations of words

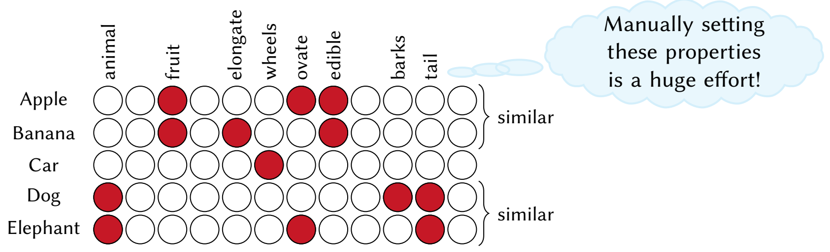
Towards word similarity

In the bag of words model, we can interpret our document vectors as:

$$\vec{d} = \sum\nolimits_{w\in d} e_w = \sum\nolimits_{w\in d} \underbrace{(0,\ldots,0,1,0,\ldots,0)}_{\text{1 only at position } w}$$

where e_w is a unit vector containing a 1 in the column corresponding to word w.

In so-called "*distributed representations*", the word information is not in a single position anymore:

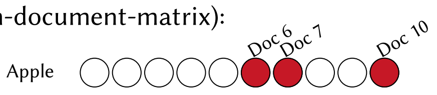


Vector representations of words

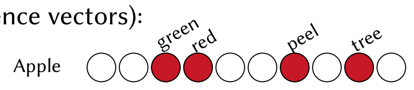
Different explicit vector representations

We can obtain such representations with different approaches:

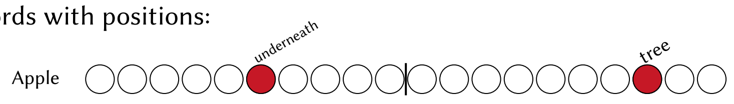
Document occurrences (term-document-matrix):



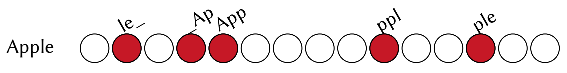
Neighboring words (cooccurrence vectors):



Neighboring words with positions:



Character trigraphs:



Vector representations of words

Learned vector representations

The previous examples were engineered, high-dimensional, and sparse features.

We can get some success with Cosine similarity to compare words.

➡ Can we *learn* lower-dimensional (dense) features from the data?

LSA can be seen as such an approach: factorize the term-document-matrix

Many methods can be seen as a variant of this:

- build a (large) explicit representation
- factorize[11] into a lower-dimensional approximation
- use approximation as new feature vector instead

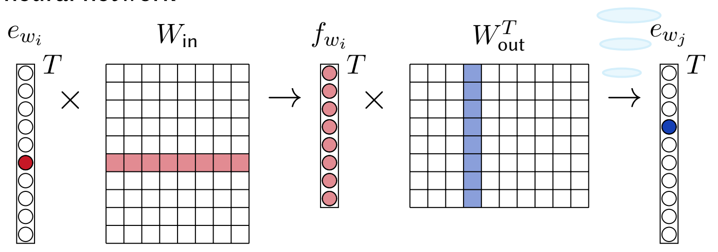
[11]Not necessarily by SVD, but instead, e.g., similar to neural networks

Neural Models for Word Similarity

Skip-Gram with Negative Sampling (SGNS, word2vec [Mik+13; LM14])

Neural networks functions: Softmax, hierarchical softmax, negative sampling

One hidden layer neural network:



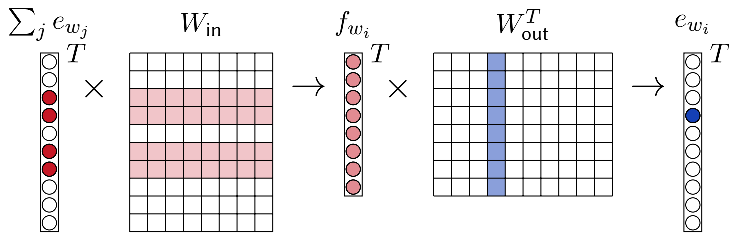
Every word corresponds to one row in the "encoder matrix" W_{in} (= word vectors).
Every word corresponds to one column in the "decoder matrix" W_{out} (usually discarded).

Weight matrixes W_{in} and W_{out} are iteratively optimized to best predict the neighbor words w_j for $j \in i-c, \dots, i-1, i+1, \dots, i+c$ and $c \approx 5$.

Neural Models for Word Similarity

Continuous Bag of Words (CBOW, word2vec [Mik+13; LM14])

One hidden layer neural network:



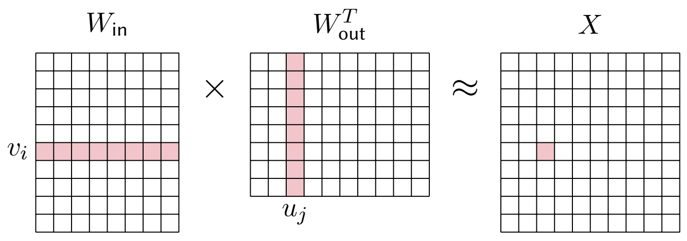
Use words $w_{i-c}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+c}$ to predict word w_i.

Intuition: sum the rows of every input word in W_{in},
find the most similar column in W_{out}^T as output.

Neural Models for Word Similarity

Interpretation [GL14; LGD15; LGD15]

For skip-gram, we can interpret $W_{\text{in}} \times W_{\text{out}}$ as follows:



Where $X = \{x_{ij}\}$ is a word cooccurrence matrix.

➡ word2vec is an (implicit) matrix factorization.

We can get similar (but not quite as good) results with SVD [LG14].

Neural Models for Word Similarity

Loss functions of word2vec [Mik+13; LM14]

Softmax

Probability of word w_j given w_i:

$$p(w_j \mid w_i) \propto \exp\left(u_j^T \cdot v_i\right) \qquad \text{such that } \sum\nolimits_j p(w_j \mid w_i) = 1$$

Loss function for Skip-Gram:

$$L_{\text{skip-gram}} = -\frac{1}{|S|} \sum\nolimits_{i \in S} \sum\nolimits_{j=-c,\dots,-1,+1,\dots,c} \log p(w_{i+j} \mid w_i)$$

Predict each word separately

Loss function for CBOW:

$$L_{\text{CBOW}} = -\frac{1}{|S|} \sum\nolimits_{i \in S} \log p\left(w_i \mid \sum\nolimits_{j=-c,\dots,-1,+1,\dots,c} w_{i+j}\right)$$

where $|S|$ is the number of context windows.

Aggregate all words

For performance, approximate softmax (testing all j is too expensive) with, e.g., hierarchical softmax or negative sampling.
Train with: backpropagation, stochastic gradient descent

Neural Models for Word Similarity

Optimizing skip-gram

We can optimize the weights using stochastic gradient descent and back-propagation.
The basic idea is to update the rows of W_{in} and W_{out} with a learning rate η:

$$w_i^{(t+1)} = w_i^{(t)} - \eta \sum\nolimits_j \epsilon_j \cdot h_j$$

where ϵ_j is the prediction error wrt. the jth target, and h_j is the jth target.

Intuitively, in each iteration we

- make the “good” output vector(s) more similar to output we computed
- make the “bad” output vector(s) less similar to output we computed
  use negative sampling: do not update all of them, only a sample
- make the input vector(s) more similar to the vector of the desired output
- make the input vector(s) less similar to the vector of the undesired output

Neural Models for Word Similarity

Global Vectors for Word Representation [PSM14]

If we aggregate all word cooccurrences into a matrix $X = \{x_{ij}\}$, the skip-gram objective:

$$L_{\text{skip-gram}} = -\frac{1}{|S|} \sum\nolimits_{i \in S} \sum\nolimits_{j=-c,\dots,-1,+1,\dots,c} \log p(w_{i+j} \mid w_i)$$

becomes

$$L_{\text{skip-gram}} = -\sum\nolimits_i \sum\nolimits_j x_{ji} \log p(w_j \mid w_i)$$

This is similar to the loss function of GloVe:

$$L_{\text{GloVe}} = -\sum\nolimits_i \sum\nolimits_j f(x_{ji}) \left(\log(x_{ij} - u_j^T \cdot v_i)\right)^2$$

weight

divergence

Neural Models for Document Similarity

From word2vec to doc2vec [LM14]

The early approaches used the average word vector, but it did not work too well.

We can design the vector representation as we like.

Idea: also include the document.

Concatenate the word vector with a document indicator $(0, \ldots, 0, 1, 0, \ldots, 0)$.

$\Rightarrow$ we also optimize a vector for each (training) document.

References I

[GL14] Y. Goldberg and O. Levy. "word2vec explained: Deriving Mikolov et al.'s negative-sampling word-embedding method". In: *CoRR* abs/1402.3722 (2014).

[Kus+15] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. "From Word Embeddings To Document Distances". In: *International Conference on Machine Learning, ICML*. 2015, pp. 957–966.

[LBB14] A. Lazaridou, E. Bruni, and M. Baroni. "Is this a wampimuk? Cross-modal mapping between distributional semantics and the visual world". In: *Annual Meeting of the Association for Computational Linguistics, ACL*. 2014, pp. 1403–1414.

[LG14] O. Levy and Y. Goldberg. "Neural Word Embedding as Implicit Matrix Factorization". In: *Neural Information Processing Systems, NIPS*. 2014, pp. 2177–2185.

[LGD15] O. Levy, Y. Goldberg, and I. Dagan. "Improving Distributional Similarity with Lessons Learned from Word Embeddings". In: *TACL* 3 (2015), pp. 211–225.

[LM14] Q. V. Le and T. Mikolov. "Distributed Representations of Sentences and Documents". In: *International Conference on Machine Learning, ICML*. 2014, pp. 1188–1196.

[Mik+13] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* abs/1301.3781 (2013).

[MR01] S. Mcdonald and M. Ramscar. "Testing the distributional hypothesis: The influence of context on judgements of semantic similarity". In: *Proceedings of the Cognitive Science Society*. 23. 2001, pp. 611–617.

[PSM14] J. Pennington, R. Socher, and C. D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing, EMNLP*. 2014, pp. 1532–1543.

Summary

Representing Documents

We learned about different ways of representing documents as vectors:

- Bag of Words (BoW), with different weights (TF-IDF)
- Topic distributions (pLSI, LDA)
- Embeddings (doc2vec)

Challenges:

- Stop words & weighting, spelling errors
- Synonyms, homonyms, negation, sarcasm, irony
- Short documents
- High dimensionality
- Similarity computations
- Evaluation

Summary

Clusters & Topics

Clusters:

- Typically every document belongs to exactly one cluster
- Soft assignment variants exist (EM, Fuzzy c-means, …)
- Often assume *dense* vectors

Frequent itemsets / frequent sequences / subspace clustering:

- A document can contain multiple frequent itemsets, or none

Topics:

- Documents are mixtures of topics
- LDA: Dirichlet prior – only a few topics per document
- Sparse vectors assumed

Summary

Recurring themes

Many methods can be interpreted as *matrix factorization*:

- Explicit, e.g., LSI/LSA
- Implicit: k-means as one-hot factorization of the feature matrix [Bau16]
- Implicit: pLSI, LDA as non-negative matrix factorization (NMF) [DLJ10]
- Implicit: word2vec as factorization of the cooccurrence matrix [LG14]

Optimization:

- k-means (Lloyds algorithm) as expectation-maximization
- EM for optimizing pLSI
- Gibbs sampling and Markov-Chain-Monte-Carlo
- Stochastic Gradient Descent

Conclusions

Text clustering / topic modeling:

- Two sides of the same coin
- Rely on *statistical* models
- Trained by numerical optimization

Usage is not trivial:

- Many parameters to choose and tune
- Many variants
- Requires fine-tuning for best results
- Hard to evaluate

➡ Do not treat these methods as a “black box” algorithms,
but try to understand how they work, so you can adapt them to your problem!

Open Problems

Many open problems – a lot of potential for thesis topics:

- Multi-lingual support (German is more difficult, and current quality can be pretty bad)
- Hierarchical topics (e.g., G20 riots as part of G20 overall)
- Temporal aspects (emerging new topics, such as the G20 riots)
- Evaluation and topic summarization
- Bias problems (“racist” algorithms, gender bias, ...)
  `man` is to `soccer` as `woman` is to ➡ `volleyball`

References I

[Bau16] C. Bauckhage. “k-Means Clustering via the Frank-Wolfe Algorithm”. In: *Lernen, Wissen, Daten, Analysen (LWDA).* 2016, pp. 311–322.

[DLJ10] C. H. Q. Ding, T. Li, and M. I. Jordan. “Convex and Semi-Nonnegative Matrix Factorizations”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 32.1 (2010), pp. 45–55.

[LG14] O. Levy and Y. Goldberg. “Neural Word Embedding as Implicit Matrix Factorization”. In: *Neural Information Processing Systems, NIPS.* 2014, pp. 2177–2185.